



PAUL SCHERRER INSTITUT

PSI-PR-08-02

The OPAL Framework

(Object Oriented Parallel Accelerator Library)

Version 1.4.0

User's Reference Manual

Andreas Adelmann, Achim Gsell, Tulin Kaman, Valeria Rizzoglio (PSI),
Christof Metzger-Kraus (HZB),
Yves Ineichen (IBM),
Xiaoying Pang, Steve Russell (LANL),
Chuan Wang, Jianjun Yang (CIAE),
Suzanne Sheehy, Chris Rogers (RAL) and
Daniel Winklehner (MIT)


Abstract

OPAL is a tool for charged-particle optics in accelerator structures and beam lines. Using the MAD language with extensions, OPAL is derived from MAD9P and is based on the CLASSIC class library, which was started in 1995 by an international collaboration. IPPL (Independent Parallel Particle Layer) is the framework which provides parallel particles and fields using data parallel ansatz. OPAL is built from the ground up as a parallel application exemplifying the fact that HPC (High Performance Computing) is the third leg of science, complementing theory and the experiment. HPC is made possible now through the increasingly sophisticated mathematical models and evolving computer power available on the desktop and in super computer centres. OPAL runs on your laptop as well as on the largest HPC clusters available today.

The OPAL framework makes it easy to add new features in the form of new C++ classes. It comes in the following flavours:

OPAL-CYCL tracks particles with 3D space charge including neighbouring turns in cyclotrons and FFAG's with time as the independent variable.

OPAL-T can be used to model beam lines, linacs, rf-photo injectors and complete XFEL's excluding the undulator.

It should be noted that not all features of OPAL are available in all flavours. The icon  OPAL-T means that a feature is not yet available in OPAL-T. Similar icons are used for the other flavours.

Contents

1	Introduction	1
1.1	Aim of OPAL and History	1
1.2	Parallel Processing Capabilities	1
1.3	Quality Management	2
1.4	Field Maps from the Femaxx 3D Eigenmode Solver	3
1.5	Output	4
1.6	Change History	5
1.7	Known Issues and Limitations	5
1.7.1	OPAL-T	5
1.7.2	OPAL-CYCL	5
1.8	Acknowledgements	5
1.9	Citation	5
2	Conventions	7
2.1	Physical Units	7
2.1.1	OPAL-cycl	8
2.2	Symbols used	8
3	Tutorial	9
3.1	Starting OPAL	9
3.2	Autophase Example	10
3.3	Examples of Particle Accelerators and Beamlines	10
3.3.1	PSI XFEL 250 MeV Injector	11
3.3.2	PSI Injector II Cyclotron	12
3.3.3	PSI Ring Cyclotron	16
3.4	Translate Old Distribution Commands to New Distribution Commands	18
3.4.1	GUNGAUSSFLATTOPTH and ASTRAFLATTOPTH Distribution Types	18
3.4.2	FROMFILE, GAUSS and BINOMIAL Distribution Types	18
4	OPAL-T	19
4.1	Introduction	19
4.2	Variables in OPAL-T	19
4.3	Integration of the Equation of Motion	20
4.4	Envelope Tracker	20
4.4.1	Envelope Equation without Dispersion For Long Beams	20
4.4.2	Slice Model	23
4.4.3	Code Comparison	27
4.5	Space Charge	27
4.5.1	FFT Based Particle-Mesh (PM) Solver	27
4.6	Wake Fields	30

4.7	Multiple Species	30
5	OPAL-CYCL	31
5.1	Introduction	31
5.2	Tracking modes	31
5.2.1	Single Particle Tracking mode	31
5.2.2	Tune Calculation mode	32
5.2.3	Multi-particle tracking mode	32
5.3	Variables in OPAL-CYCL	32
5.3.1	The initial distribution in the local reference frame	33
5.4	Field Maps	33
5.4.1	CARBONCYCL type	34
5.4.2	CYCIAE type	35
5.4.3	BANDRF type	36
5.4.4	Default PSI format	36
5.4.5	3D fieldmap	36
5.4.6	user's own fieldmap	37
5.5	RF field	37
5.5.1	Read RF voltage profile	37
5.5.2	Read 3D RF fieldmap	37
5.6	Particle Tracking and Acceleration	38
5.7	Space Charge	38
5.8	Multi-bunches Issues	38
5.9	Input	39
5.10	Output	40
6	Command Format	41
6.1	Statements and Comments	41
6.2	Identifiers or Labels	42
6.3	Command Attribute Types	42
6.4	String Attributes	43
6.5	Logical Expressions	44
6.6	Real Expressions	45
6.7	Operators	45
6.8	Operands in Expressions	46
6.8.1	Literal Constants	46
6.8.2	Symbolic constants	47
6.8.3	Variable labels	48
6.8.4	Element or command attributes	49
6.8.5	Deferred Expressions and Random Values	49
6.9	Element Selection	49
6.9.1	Element Selection	49
6.9.2	Range Selection	50
6.10	Constraints	51
6.11	Variable Names	51
6.12	Regular Expressions	51
6.13	Arrays	52
6.13.1	Logical Arrays	52
6.13.2	Real Arrays	53
6.13.3	String Arrays	55
6.13.4	Token List Arrays	55

7	Control Statements	57
7.1	Getting Help	57
7.1.1	HELP Command	57
7.1.2	SHOW Command	57
7.1.3	WHAT Command	57
7.2	STOP / QUIT Statement	58
7.3	OPTION Statement	58
7.4	Parameter Statements	60
7.4.1	Variable Definitions	60
7.4.2	Symbolic Constants	62
7.4.3	Vector Values	63
7.4.4	Assignment to Variables	63
7.4.5	VALUE: Output of Expressions	63
7.5	Miscellaneous Commands	64
7.5.1	ECHO Statement	64
7.5.2	SYSTEM: Execute System Command	64
7.5.3	SYSTEM Command under UNIX	64
7.6	TITLE Statement	64
7.7	File Handling	65
7.7.1	CALL Statement	65
7.7.2	SAVE Statement	65
7.8	IF: Conditional Execution	65
7.9	WHILE: Repeated Execution	66
7.10	MACRO: Macro Statements (Subroutines)	66
8	Elements	67
8.1	Element Input Format	67
8.2	Common Attributes for all Elements	67
8.3	Drift Spaces	69
8.4	Bending Magnets	70
8.4.1	RBend (OPAL-T)	70
8.4.2	SBend (OPAL-T)	73
8.4.3	RBend and SBend Examples (OPAL-T)	75
8.4.4	Bend Fields from 1D Field Maps (OPAL-T)	82
8.4.5	Default Field Map (OPAL-T)	85
8.4.6	SBend3D (OPAL-CYCL)	87
8.5	Quadrupole	88
8.6	Sextupole	89
8.7	Octupole	89
8.8	General Multipole	90
8.9	Solenoid	90
8.10	Cyclotron	91
8.11	RingDefinition	93
8.11.1	Local Cartesian Offset	93
8.12	RF Cavities (OPAL-T and OPAL-CYCL)	94
8.12.1	OPAL-T mode	94
8.12.2	OPAL-CYCL mode	94
8.13	RF Cavities with Time Dependent Parameters	95
8.13.1	Time Dependence	96
8.13.2	Example	96
8.14	Traveling Wave Structure	97

8.15	Monitors	98
8.16	Collimators	100
8.16.1	OPAL-T mode	100
8.16.2	OPAL-CYCL mode	101
8.17	Septum (OPAL-CYCL)	102
8.18	Probe (OPAL-CYCL)	102
8.19	Stripper (OPAL-CYCL)	103
8.20	Degrader (OPAL-T)	104
8.21	Correctors (OPAL-T)	104
9	Beam Lines	105
9.1	Simple Beam Lines	105
9.2	Sub-lines	105
10	Physics Commands	107
10.1	BEAM Command	107
11	Distribution Command	109
11.1	Units	110
11.2	General Distribution Attributes	110
11.2.1	Universal Attributes	112
11.2.2	Injected Distribution Attributes	113
11.2.3	Emitted Distribution Attributes	113
11.3	FROMFILE Distribution Type	114
11.4	GAUSS Distribution Type	115
11.4.1	Simple GAUSS Distribution Type	115
11.4.2	GAUSS Distribution for Photoinjector	117
11.4.3	Correlations for GAUSS Distribution (Experimental)	119
11.5	FLATTOP Distribution Type	120
11.5.1	Injected FLATTOP	120
11.5.2	Emitted FLATTOP	121
11.5.3	Transverse Distribution from Laser Profile (Under Development)	122
11.5.4	GUNGAUSSFLATTOP Distribution Type	122
11.5.5	ASTRAFLATTOP Distribution Type	122
11.6	BINOMIAL Distribution Type	122
11.7	Emission Models	122
11.7.1	Emission Model: NONE (default)	123
11.7.2	Emission Model: ASTRA	123
11.7.3	Emission Model: NONEQUIL	124
11.8	Distribution List	124
12	Fieldsolver	127
12.1	Fieldsolver Command	127
12.2	Define the Fieldsolver to be used	127
12.3	Define Domain Decomposition	127
12.4	Define Number of Gridpoints	127
12.5	Define Boundary Conditions	127
12.6	Define Greens Function	127
12.7	Define Bounding Box Enlargement	128
12.8	Define Geometry	128
12.9	Define Iterative Solver	128
12.10	Define Interpolation for Boundary Points	129

12.11 Define Tolerance	129
12.12 Define Maximal Iterations	129
12.13 Define Preconditioner Behaviour	129
12.14 Define the number of Energy Bins to use	129
13 Wakefields	131
13.1 Wakefield Command	132
13.2 Define the Wakefield to be used	132
13.3 Define the wakefield type	132
13.4 Define the number of bins	133
13.5 Define the bunch length to be constant	133
13.6 Define the conductivity	133
13.7 Define the impedance	133
13.8 Define the form of the beam pipe	133
13.9 Define the radius of the beam pipe	133
13.10 Define the σ of the beam pipe	133
13.11 Define the relaxation time (τ) of the beam pipe	133
13.12 Import a wakefield from a file	133
13.13 Wake Functions	134
13.14 Filters	134
14 Geometry	137
14.1 Geometry Command	137
14.2 Define the Geometry File	137
14.3 Define the Length	137
14.4 Define the Start	138
14.5 Define the Semi-Major Axis	138
14.6 Define the Semi-Minor Axis	138
15 Tracking	139
15.1 Track Mode	139
15.1.1 Track a Random Machine	141
16 Field Emission	143
16.1 Field Emission Command	144
17 Multipacting	145
17.1 Commands Related to Multipacting Simulation	148
17.2 Run Parallel Plate Benchmark	151
17.3 PostProcessing	153
18 Physics Models Used in the Particle Matter Interaction Model	155
18.1 The Energy Loss	155
18.2 The Coulomb Scattering	156
18.2.1 Multiple Coulomb Scattering	156
18.2.2 Large Angle Rutherford Scattering	156
18.3 The Flow Diagram of <i>CollimatorPhysics</i> Class in OPAL	157
18.3.1 The Substeps	160
18.4 Available Materials in OPAL	160
18.5 Example of an Input File	160
18.6 A Simple Test	161

A	Installation	163
A.1	Build and install OPAL on a Mac & Linux	163
A.1.1	Supporting Libraries	163
A.1.2	Environment Variables	163
A.1.3	Installing OPAL	165
A.2	Cray XE6 Installation – outdated	165
A.2.1	.bash_profile.ext File	165
A.2.2	.bashrc.ext File	165
A.2.3	OPAL	166
A.3	Build unit tests	167
A.4	Using pre-build Binaries	168
A.5	Enabling the Iterative Solvers for Space-Charge Calculation	168
A.5.1	Environment Variables	169
A.5.2	Build OPAL with SAAMG_SOLVER enabled	170
A.5.3	Build OPAL with AMR_SOLVER enabled	170
A.6	Debug Flags	171
A.7	OPAL as a Library	172
A.8	Examples	172
B	OPAL Language Syntax	173
C	OPAL-T Field Maps	183
C.1	Introduction	183
C.2	Comments in Field Maps	183
C.3	Field Map Warnings and Errors	183
C.4	Types and Format	185
C.5	Field Map Orientation	186
C.6	FAST Attribute for 1D Field Maps	187
C.7	1DMagnetoStatic	189
C.8	AstraMagnetostatic	191
C.9	1DDynamic	193
C.10	AstraDynamic	195
C.11	1DProfile1	197
C.11.1	1DProfile1 Type 1 for Bend Magnet	203
C.11.2	1DProfile1 Type 2 for Bend Magnet	204
C.12	2DElectroStatic	206
C.13	2DMagnetoStatic	208
C.14	2DDynamic	210
C.15	3DDynamic	212
D	OPAL - MADX Conversion Guide	215
E	Autophasing Algorithm	217
E.1	Standing Wave Cavity	217
E.2	Traveling Wave Structure	218
E.2.1	Alternative Approache for Traveling Wave Structures	219

F	Benchmarks	223
F.1	OPAL-T compared with TRANSPORT & TRACE 3D	223
F.1.1	TRACE 3D	223
F.1.2	TRACE 3D Units	223
F.1.3	TRACE 3D Input beam	224
F.1.4	TRANSPORT	225
F.1.5	TRANSPORT Units	225
F.1.6	TRANSPORT Input beam	225
F.1.7	Comparison TRACE 3D and TRANSPORT	226
F.1.8	Relations to OPAL-T	233
F.1.9	OPAL-T Units	233
F.1.10	OPAL-T Input beam	233
F.1.11	Comparison TRACE 3D and OPAL-T	234
F.1.12	Conclusion	237
F.2	Hard Edge Dipole Comparison with ELEGANT	238
F.2.1	OPAL Dipole	238
F.2.2	Map for Hard Edge Dipole	238
F.2.3	Integration Time Step	239
F.3	1D CSR comparison with ELEGANT	239
F.3.1	Benchmark	241
F.4	OPAL & IMPACT-T	
	impactt	243
F.4.1	OPAL Input	243
F.4.2	IMPACT-T Input	243
F.4.3	Results	245
G	Changelog	247
G.1	Changes in OPAL Version 1.3.0	247
G.1.1	Cyclotron Element	247
G.1.2	AMG Solver	247
G.1.3	Variable Frequency RF	247
G.1.4	Appendix	247
G.1.5	Unit Tests	247
G.2	Changes in OPAL Version 1.2.0	247
G.2.1	Distribution	247
G.2.2	SBEND and RBEND	248
G.2.3	Misalignment	248
G.2.4	New elements: SBend3D & RINGDEFINITION	248
G.2.5	Space Charge Solver	248

List of Tables

2.1	Physical Units	7
6.1	String Operator in OPAL	44
6.2	String Function in OPAL	44
6.3	Logical Operators in OPAL	44
6.4	Real Operators in OPAL	46
6.5	Real Functions in OPAL	46
6.6	Real Functions with one in OPAL	47
6.7	Real Functions of Arrays in OPAL	48
6.8	Predefined Symbolic Constants	48
6.9	Real Array Functions in OPAL (acting component-wise)	54
7.1	Default Settings for Options	61
11.1	Possible distribution types. Note that the SURFACEEMISSION and SURFACERANDCREATE distribution types will not be discussed in this chapter. Instead, refer to Chapter 16 on field emission for using these types.	109
11.2	Definition of INPUTMOUNTS attribute.	110
11.3	Definition of EMITTED attribute.	111
11.4	Definition of universal distribution attributes. Any distribution type can use these and they are the same whether the beam is <i>injected</i> or <i>emitted</i>	112
11.5	Definition of distribution attributes that only affect <i>injected</i> beams.	113
11.6	Definition of distribution attributes that only affect <i>emitted</i> beams.	113
11.7	Definition of distribution attributes for a FROMFILE distribution type.	114
11.8	File format for <i>injected</i> FROMFILE distribution type. N is the number of particles in the file. The particle coordinates are defined in Sections 4.2 or 5.3.	114
11.9	File format for <i>emitted</i> FROMFILE distribution type. N is the number of particles in the file. The particle coordinates are defined in Section 4.2 except that z , in meters, is replaced by t in seconds.	115
11.10	Definition of the basic distribution attributes for a GAUSS distribution type.	116
11.11	Definition of additional distribution attributes for an <i>emitted</i> GAUSS distribution type. These are used to generate a distribution with a time profile as illustrated in Figure 11.1.	117
11.12	Definition of additional distribution attributes for a GAUSS distribution type for generating correlations in the beam.	119
11.13	Definition of the basic distribution attributes for an <i>injected</i> FLATTOP distribution type.	120
11.14	Definition of the basic distribution attributes for an <i>emitted</i> FLATTOP distribution type.	121
11.15	Different distributions specified by a single parameter m	122
11.16	Attributes for the NONE and ASTRA emission models.	123
11.17	Attributes for the NONE and ASTRA emission models.	124
12.1	Fieldsolver command summary	128

12.2	Preconditioner behaviour command summary	129
13.1	Wakefield command summary	132
14.1	Geometry command summary	137
15.1	Commands accepted in Tracking Mode	139
16.1	Field Emission Command summary	144
17.1	Multipacting Related Command Summary	150
18.1	List of materials with their parameters implemented in OPAL	160
A.1	Debug flags.	171
C.1	Layout of a 1DMagnetoStatic field map file.	189
C.2	Layout of an AstraMagnetoStatic field map file.	191
C.3	Layout of a 1DDynamic field map file.	193
C.4	Layout of an AstraDynamic field map file.	195
C.5	Layout of a 1DProfile1 field map file.	200
C.6	Layout of a 2DElectroStatic field map file.	206
C.7	Layout of a 2DMagnetoStatic field map file.	208
C.8	Layout of a 2DDynamic field map file.	210
C.9	Layout of a 3DDynamic field map file.	212
F.1	Bending magnet description in TRACE 3D and values used in the simulation	228
F.2	Edge angle description in TRACE 3D and values used in the simulation	229
F.3	Bending magnet description in TRANSPORT and values used in the simulation	229
F.4	Edge angle and fringe field description in TRANSPORT and values used in the simulation	231
F.5	Transversal beam size at the end of each element in the line printed out by TRACE 3D and TRANSPORT	231
F.6	Horizontal and longitudinal emittance comparison between TRACE 3D and TRANSPORT, both expressed in π -mm-mrad	232
F.7	Bending magnet features in TRACE 3D and TRANSPORT	232
F.8	Main features of the three beam dynamics codes: TRACE 3D, TRANSPORT and OPAL	233
F.9	Bending magnet features in TRACE 3D and OPAL-T	237

List of Figures

1.1	Parallel efficiency and particles pushed per μs as a function of cores	2
1.2	Comparison of energy and emittance in x between IMPACT-Tand OPAL-T	3
1.3	Tetrahedral mesh of a pillbox shaped cavity	4
3.1	Reference orbit(left) and tune diagram(right) in Injector II	14
3.2	Radial and vertical eigenellipse at 2 MeV of Injector II	15
3.3	Energy Vs. time (left) and external B field Vs. trackstep (Right, only show for about 2 turns) . . .	17
3.4	Vertical phase at different energy from left to right: 0.87 MeV, 15 MeV and 35 MeV	17
3.5	Reference orbit(left) and tune diagram(right) in Ring cyclotron	17
4.1	Evolution of the particle beam in the multi-slice approximation.	23
4.2	The bunch form factor $g(\zeta) = G(\zeta, A)/2$	26
4.3	Comparison of the OPAL-e envelope tracker and the full 3D OPAL-t tracker in the first 12 meters fof the 250 MeV test injector at PSI. Shown here are both σ_x and σ_z (a), and the emittance (b). .	27
5.1	2D field map on the median plane with primary direction corresponding to the azimuthal direction, secondary direction to the radial direction	34
8.1	Illustration of a general rectangular bend (RBEND) with a positive bend angle α . The entrance edge angle, E_1 , is positive in this example. An RBEND has parallel entrance and exit pole faces, so the exit angle, E_2 , is uniquely determined by the bend angle, α , and E_1 ($E_2 = \alpha - E_1$). For a positively charge particle, the magnetic field is directed out of the page.	70
8.2	Visualisation of angles used to rotate the bend relative to the incoming beam, where \mathbf{n} is the normal of the face.	72
8.3	Illustration of a general sector bend (SBEND) with a positive bend angle α . In this example the entrance and exit edge angles E_1 and E_2 have positive values. For a positively charge particle, the magnetic field is directed out of the page.	73
8.4	Plot of the entrance fringe field of a dipole magnet along the mid-plane, perpendicular to its entrance face. The field is normalized to 1.0. In this case, the fringe field is described by an Enge function (see Equation 8.1) with the parameters from the default <code>1DProfile1</code> field map described in 8.4.5. The exit fringe field of this magnet is the mirror image.	83
8.5	Schematic of the simplified geometry of a cavity gap and parameters	95
8.6	The on-axis field of an S-band TRAVELINGWAVE structure	97
11.1	OPAL emitted GAUSS distribution with flat top.	118
17.1	Typical SEY curve	146
17.2	Sketch map of the secondary emission process.	146

17.3	Time evolution of electron number predicted by theoretical model and OPAL simulation using Furman-Pivi's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 200MHz$, $V_0 = 120V$, $d = 5mm$	151
17.4	Time evolution of electron number predicted by theoretical model and OPAL simulation using Vaughan's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 1640MHz$, $V_0 = 120V$, $d = 1mm$	151
18.1	The comparison of stopping power with PSTAR.	155
18.2	The comparison of Coulomb scattering with Jackson's book.	157
18.3	The diagram of CollimatorPhysics in OPAL.	158
18.4	The diagram of CollimatorPhysics in OPAL(Continued).	159
18.5	The passage of protons through the collimator.	161
18.6	The energy spectrum and scattering angle at $z=0.1$ m	162
C.2	The longitudinal phase space after a gun simulation using a 1D field map (on-axis field) of the gun, a 1D field map (on-axis field) of the gun in combination with the FAST switch, and a 2D field map of the gun generated by Superfish [43].	188
C.3	Example of a 1DMagnetoStatic field map	189
C.4	Example of an ASTRA compatible magnetostatic field map	191
C.5	Example of a 1DDynamic field map	193
C.6	Example of an ASTRA compatible dynamic field map	195
C.7	Example of Enge functions describing the entrance and exit fringe fields of a rectangular bend magnet. The top part of the figure shows the relative field strength on the mid-plane. The bottom part of the figure shows an example of a particle trajectory through the magnet. Note that the magnet field is naturally divided into three regions: entrance fringe field, central field, and exit fringe field.	198
C.8	Illustration of a rectangular bend (RBEND see 8.4.1) showing the entrance and exit fringe field regions. Δ_1 is the perpendicular distance in front of the entrance edge of the magnet where the magnet fringe fields are non-negligible. Δ_2 is the perpendicular distance behind the entrance edge of the magnet where the entrance Enge function stops being used to calculate the magnet field. The reference trajectory entrance point is indicated by $O_{entrance}$. Δ_3 is the perpendicular distance in front of the exit edge of the magnet where the exit Enge function starts being used to calculate the magnet field. (In the region between Δ_2 and Δ_3 the field of the magnet is a constant value.) Δ_4 is the perpendicular distance after the exit edge of the magnet where the magnet fringe fields are non-negligible. The reference trajectory exit point is indicated by O_{exit}	201
C.9	Illustration of a sector bend (SBEND see 8.4.2) showing the entrance and exit fringe field regions. Δ_1 is the perpendicular distance in front of the entrance edge of the magnet where the magnet fringe fields are non-negligible. Δ_2 is the perpendicular distance behind the entrance edge of the magnet where the entrance Enge function stops being used to calculate the magnet field. The reference trajectory entrance point is indicated by $O_{entrance}$. Δ_3 is the perpendicular distance in front of the exit edge of the magnet where the exit Enge function starts being used to calculate the magnet field. (In the region between Δ_2 and Δ_3 the field of the magnet is a constant value.) Δ_4 is the perpendicular distance after the exit edge of the magnet where the magnet fringe fields are non-negligible. The reference trajectory exit point is indicated by O_{exit}	202
C.10	Example of a 1DProfile1 Type 1 field map	203
C.11	Example of a 1DProfile1 Type 2 field map	205
C.12	Example of a 2DElectroStatic field map	206
C.13	Example of a 2DMagnetoStatic field map	208
C.14	Example of a 2DDynamic field map	210
C.15	Example of a 3DDynamic field map	212

E.1	Field map 'FINLB02-RAC.T7' of type 1DDynamic	219
F.1	TRACE 3D graphic interface where: (1) input beam in transverse plane (above) and longitudinal plane (below); (2) output beam in transverse plane (above) and longitudinal plane (below); (3) summary of beam parameters such as input and output emittances and desired value for matching function; (4) line lattice with different elements and beam envelope. The color legend is: blu line for horizontal plane, red line for vertical plane, green line for longitudinal plane and yellow line for dispersion.	224
F.2	Sigma-matrix structure in TRACE 3D [68]	226
F.3	GUI TRANSPORT graphic interface [70]. The continuous lines describe the beam envelope in the vertical plane (above) and horizontal plane (below). The dashed line displays the dispersion. The elements in the beam line are drawn as blue and red rectangles	226
F.4	TRACE 3D input beam in the transversal plane (above) and in the longitudinal plane (below) . .	227
F.5	TRACE 3D sigma-matrix for the input beam	227
F.6	Beam envelopes in TRACE 3D for a SBEND with entrance and exit edge angles. The blue line describes the beam envelope in the horizontal plane, the red line in the vertical plane, the green line in the longitudinal plane. The yellow line displays the dispersion	228
F.7	Beam envelopes in TRANSPORT for a SBEND with entrance and exit edge angles. The continuous lines describe the beam envelope in the vertical plane (above) and horizontal plane (below). The dashed line displays the dispersion.	230
F.8	Transversal beam size comparison between TRACE 3D and TRANSPORT	230
F.9	Emittance comparison between TRACE and TRANSPORT	231
F.10	TRACE 3D and OPAL comparison: SBEND without edge angles	234
F.11	TRACE 3D and OPAL comparison: SBEND with edge angles	235
F.12	TRACE 3D and OPAL comparison: SBEND with field index and default field map	236
F.13	TRACE 3D and OPAL comparison: SBEND with field index and test field map	236
F.14	Compare emittances and beam sizes obtained by using the hard edge map (OPAL), the default map (OPAL), and the ELEGANT	239
F.15	Horizontal and vertical normalised emittances for different integration time steps	240
F.16	Normalised horizontal emittance for different fringe field ranges and integration time steps . . .	240
F.17	Zoom in on the final emittance in Figure F.17	241
F.18	Comparison of the trace space using ELEGANT and OPAL	242
F.19	$\frac{\Delta p}{p}$ in Elegant and OPAL	242
F.20	Transverse emittances in ELEGANT and OPAL	242
F.21	Transverse beam sizes and emittances in IMPACT-T and OPAL	245
F.22	Longitudinal beam size and emittance in IMPACT-T and OPAL	245

Chapter 1

Introduction

1.1 Aim of OPAL and History

OPAL is a tool for charged-particle optics in accelerator structures and beam lines. Using the MAD language with extensions, OPAL is derived from MAD9P and is based on the **CLASSIC** class library, which was started in 1995 by an international collaboration. IPPL (Independent Parallel Particle Layer) is the framework which provides parallel particles and fields using data parallel ansatz. OPAL is built from the ground up as a parallel application exemplifying the fact that HPC (High Performance Computing) is the third leg of science, complementing theory and the experiment. HPC is made possible now through the increasingly sophisticated mathematical models and evolving computer power available on the desktop and in super computer centers. OPAL runs on your laptop as well as on the largest HPC clusters available today.

The OPAL framework makes it easy to add new features in the form of new C++ classes.

OPAL comes in the following flavours:


- OPAL-CYCL
- OPAL-T
- OPAL-E

OPAL-MAP tracks particles with 3D space charge using split operator techniques, and is a proper subset of MAD9P. In the future a linear space charge mode will become available allowing the user to track moments of the distribution.

OPAL-CYCL tracks particles with 3D space charge including neighbouring turns in cyclotrons with time as the independent variable.

OPAL-T is a superset of IMPACT-T [39] and can be used to model guns, injectors and complete XFEL's excluding the undulator.

It should be noted that not all features of OPAL are available in all flavours.

The following icon  OPAL-T means that a feature is not yet available in OPAL-T. Similar icons are used for the other flavours.

1.2 Parallel Processing Capabilities

OPAL is built to harness the power of parallel processing for an improved quantitative understanding of particle accelerators. This goal can only be achieved with detailed 3D modelling capabilities and a sufficient number of simulation particles to obtain meaningful statistics on various quantities of the particle ensemble such as emittance, slice emittance, halo extension etc.

The following example is exemplifying this fact:

Table 1.1: Parameters Parallel Performance Example

Distribution	Particles	Mesh	Greens Function	Time steps
Gauss 3D	10^8	1024^3	Integrated	10

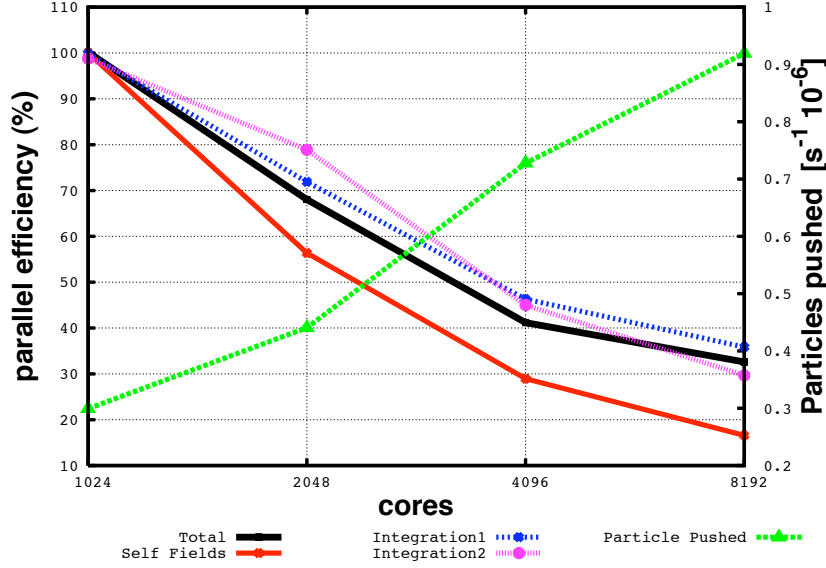
Figure 1.1: Parallel efficiency and particles pushed per μs as a function of cores

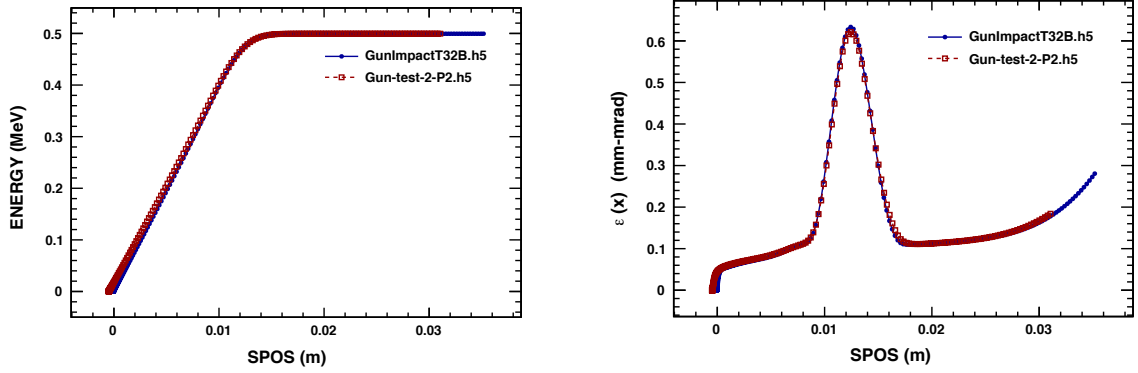
Figure 1.1 shows the parallel efficiency time as a function of used cores for a test example with parameters given in Tab. 1.1. The data were obtained on a Cray XT5 at the Swiss Center for Scientific Computing.

1.3 Quality Management

Documentation and quality assurance are given our highest attention since we are convinced that adequate documentation is a key factor in the usefulness of a code like OPAL to study present and future particle accelerators. Using tools such as a source code version control system ([git](#)), source code documentation (Doxygen) (check out, for instance, [ParallelTTracker](#)) and the extensive user manual you are now enjoying, we are committed to providing users as well as co-developers with state-of-the-art documentation to OPAL.

One example of a non-trivial test-example is the PSI DC GUN. In Figure 1.2 the comparison between IMPACT-T and OPAL-T is shown. This example is part of the regression test suite that is run every night. The input file is found in Section 3.3.

Misprints and obscurity are almost inevitable in a document of this size. Comments and *active contributions* from readers are therefore most welcome. They may be sent to andreas.adelmann@psi.ch.

Figure 1.2: Comparison of energy and emittance in x between IMPACT-Tand OPAL-T

1.4 Field Maps from the Femaxx 3D Eigenmode Solver

[TODO: AA will rewrite]

Electromagnetic field maps for beam dynamics calculations originate from a number of different electromagnetic solvers, e.g. Superfish and similar codes.

Here, we describe the current status of work in progress which will, eventually, allow the usage of field maps that have been computed with the femaxx 3-dimensional electromagnetic eigenmodal solver [47, 48].

The femaxx code computes electromagnetic eigenmodes of resonant cavities of arbitrary 3-dimensional shape and boundary conditions.

Unlike Superfish and similar 2-dimensional codes, femaxx is not restricted in the kind of geometry it can model. It is therefore possible to consider arbitrary shapes and their inclusion in beam dynamics and particle tracking calculations.

Given a mesh of a 3-dimensional geometry femaxx computes eigenomdal field decompositions.

The user then specifies sampling locations for the electromagnetic eigenfields.

At present, sampling locations are specified in terms of a cylinder shape, i.e. the user indicates the cylinder axis, the radial cylinder vector and the number of sampling locations in axial, radial and azimuthal directions.

Once the eigenmodal solution has been computed the fields are sampled at these locations and stored in the T7 file format, for subsequent use in OPAL.

Considerable effort has been spent for the validation and benchmarking of beam dynamics calculations based on T7 field maps computed with femaxx.

A pillbox cavity, i.e. a cylinder shape with a radius $r = 4.7\text{cm}$ and height $h = 3\text{ cm}$, has been chosen for benchmarking purposes, due to the availability of an analytical solution.

The analytical resonance frequency of the dominant mode is 2.441 GHz.

We have compared two cases with OPAL: (1) The analytical solution has been sampled within a cylinder volume, stored into a T7 file and used in an OPAL run; (2) the same pillbox shaped geometry has been discretization into tetrahedra and the eigenmodal fields were calculated with femaxx. These two cases were then compared, resulting in the following conclusions: (1) Using a relatively coarse mesh with some 110'000 tetrahedra, the difference between the analytical and the numerical solution was usually smaller than 1 percent. (2) Using an adaptively refined mesh, the difference between analytical and numerical solutions decreased below 1 pro mille. The mesh is shown in the figure (1.3). (3) It is therefore imperative to usa a tetrahedral mesh which has been refined around the beam axis. It is definitely more efficient to use local refinement, based on physical argument, than simply refine the complete mesh in a uniform manner.

We are now working towards benchmarking more complicated shapes in order to assess requirements w.r.t to meshes and modeling geometry so that we achieve the same or better accuracy as has been obtained from field maps that were computed with Superfish like solvers based on azimuthal symmetry.

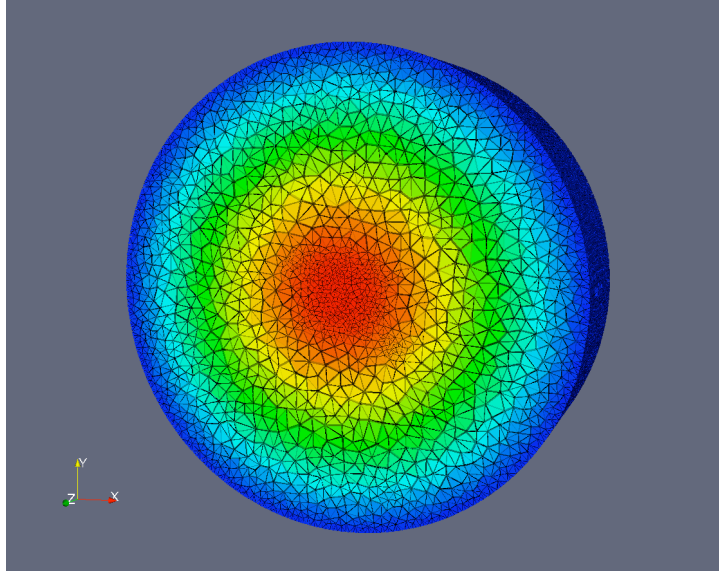


Figure 1.3: We show the discretization of a pillbox shaped cavity geometry into a tetrahedral mesh. The mesh has been adaptively refined so that the region around the cylinder axis is decomposed into smaller tetrahedra than those which are further away from the axis.

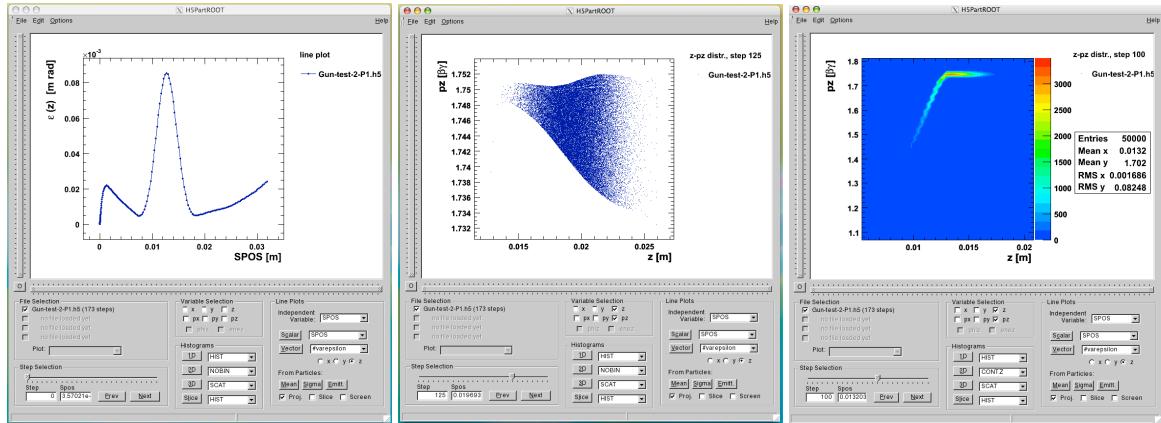


Figure 1.4: H5root enables a variety of data analysis and post processing task on OPAL data

1.5 Output

The phase space is stored in the H5hut file-format (<https://amas.psi.ch/H5hut>) and can be analysed using e.g. H5root (<http://amas.web.psi.ch/tools/H5root/index.html>). The frequency of the data output (phase space and some statistical quantities) can be controlled using the `OPTION` statement (see §7.3), with the flag `PSDUMPFREQ`. The file is named like in input file but with the extension `.h5`.

A SDDS compatible ASCII file with statistical beam parameters is written to a file with extension `.stat`. The frequency with which this data is written can be controlled with the `OPTION` statement (see §7.3) with the flag `STATDUMPFREQ`.

1.6 Change History

See Appendix G for a detailed list of changes in OPAL.

1.7 Known Issues and Limitations

1.7.1 OPAL-T

- Fields of bends may not overlap with any other field.

1.7.2 OPAL-CYCL

- Restart with the option PSDUMPLOCALFRAME does not work yet,
- In complicated geometries such as spiral inflectors, proper particle deletion at the boundary sometimes fails.

1.8 Acknowledgements

The contributions of various individuals and groups are acknowledged in the relevant chapters, however a few individuals have or had considerable influence on the development of OPAL, namely Chris Iselin, John Jowett, Julian Cummings, Ji Qiang, Robert Ryne and Stefan Adam. For the H5ROOT visualization tool credits go to Thomas Schietinger. The effort to couple FEMAXX to OPAL was led by Benedikt Oswald.

The following individuals are acknowledged for past contributions: Yuanjie Bi, Colwyn Gulliford, Hao Zha and Christopher Mayes.

1.9 Citation

Please cite OPAL in the following way:

```
@techreport{opal:1,
title="{The OPAL (Object Oriented Parallel Accelerator Library) Framework}",
author="Andreas Adelmann and Achim Gsell and Tulin Kaman (PSI) and Christof Metzger-Kraus
(HZB) and Yves Ineichen (IBM) and Steve Russell (LANL) and Chuan Wang and Jianjun Yang (CIAE)
and Suzanne Sheehy and Chris Rogers (RAL) and Daniel Winklehner (MIT)",
institution="Paul Scherrer Institut",
number="PSI-PR-08-02",
year="(2008-2014) "
}
```

Chapter 2

Conventions

2.1 Physical Units

Throughout the computations OPAL uses international units (see Tab. 2.1), as defined by SI (Système International).

Table 2.1: Physical Units

Quantity	Dimension
Length	m (metres)
Angle	rad (radians)
Quadrupole coefficient	m^{-2}
Multipole coefficient, 2n poles	m^{-n}
Electric voltage	MV (Megavolts)
Electric field strength	MV/m
Frequency	MHz (Megahertz)
Particle energy	GeV or eV
Particle mass	GeV/c^2
Particle momentum	$\beta\gamma$ od eV
Beam current	A (Amperes)
Particle charge	e (elementary charges)
Impedances	$\text{M}\Omega$ (Megohms)
Emittances	m rad^1
RF power	MW (Megawatts)

2.1.1 OPAL-cycl

The OPAL-cycl flavour (Sec. 8.10) is using the so called Cyclotron units. Lengths are define in (mm) frequencies in (MHz) momenta in ($\beta\gamma$) and angles in (deg), except **RFPHI** which is in (rad).

2.2 Symbols used

X	Ellipse axis along the x dimension [m]. $X = R$ for circular beams.
Y	Ellipse axis along the y dimension [m]. $Y = R$ for circular beams.
R	Beam radius for circular beam [m].
R^*	Effective beam radius for elliptical beam: $R^* = (X + Y)/2$ [m].
σ_x	Rms beam size in x : $\sigma_x = \langle x^2 \rangle^{1/2}$ [m]. $\sigma_x = X/2$ for elliptical or circular beams ($X=Y=R$).
σ_y	Rms beam size in y : $\sigma_y = \langle y^2 \rangle^{1/2}$ [m]. $\sigma_y = Y/2$ for elliptical or circular beams ($X=Y=R$).
σ_i	Rms beam size in x ($i=1$) or y ($i=2$): $\sigma = \langle x^2 \rangle^{1/2}$ or $\langle y^2 \rangle^{1/2}$ [m].
σ_L	Rms beam size in the Larmor frame for cylindrically symmetric beam and external fields [m]: $\sigma_L = \sigma_x = \sigma_y$.
σ_r	Rms beam size in r for a circular beam: $\sigma_r = \langle r^2 \rangle^{1/2} = R/\sqrt{2}$ [m].
σ^*	Average rms size for elliptical beam: $\sigma^* = (\sigma_x + \sigma_y)/2$ [m].
θ_r	Larmor angle [rad]
$\dot{\theta}_r$	Time derivative of Larmor angle: $\dot{\theta}_r = -eB_z/2m\gamma$ [rad/sec].
z_s	Longitudinal position of a particular beam slice [m].
z_h, z_t	Position of the head & tail of a beam bunch [m].
ζ	Used to label the position of a beam slice in the beam [m]. For bunched beams: $\zeta = z_s - z_t$.
ξ	Used to label the position of a slice image charge [m]. For bunched beams: $\xi = z_h + z_t$.
K	Focusing function of cylindrically symmetric external fields: $K = -\frac{\partial F_r}{\partial r}$ [N/m].
K_i	Focusing function in x_i direction: $K_i = -\frac{\partial F_{x_i}}{\partial x_i}$ [N/m].
I_0	Alfven current: $I_0 = e/4\pi\epsilon_0 mc^3$ [A].
I	Beam current [A].
$I(\zeta)$	Slice beam current [A].
k_p	Beam perveance: $k_p = I(\zeta)/2I_0$
$g(\zeta)$	Form factor used in slice analysis of bunched beams.

Chapter 3

Tutorial

This chapter will provide a jump start describing some of the most common used features of OPAL. The complete set of examples can be found and downloaded at <https://amas.psi.ch/OPAL/wiki/OPAL>. All examples run on a single core, but can be used efficiently on up to 8 cores. OPAL scales in the weak sense, hence for a higher concurrency one has to increase the problem size i.e. number of macro particles and the grid size, which is beyond this tutorial.

3.1 Starting OPAL

The name of the application is `opal`. When called without any argument an interactive session is started.

```
\$ opal
Ippl> CommMPI: Parent process waiting for children ...
Ippl> CommMPI: Initialization complete.
>
>      /_/_\  /_/_\  /\  | |
>      | | | | |_) / \  | |
>      | | | | |_/  \ \ | |
>      | | | | | /_/_\ | |
>      \_/_/ | | /_/_\ \_/_/
OPAL >
OPAL > This is OPAL (Object Oriented Parallel Accelerator Library) Version 1.3.0rc1 ...
OPAL >
OPAL > Please send cookies, goodies or other motivations (wine and beer ... ) to ...
OPAL > Time: 16.43.23 date: 26/04/2013
OPAL > Reading startup file ``/Users/adelmann/init.opal``.
OPAL > Finished reading startup file.
==>
```

One can exit from this session with the command `QUIT;` (including the semicolon).

For batch runs OPAL accepts the following command line arguments:

Argument	Values	Function
--info	0 – 5	controls the amount of output to the command line. 0 means no or scarce output, 5 means a lot of output. Default: 1. Full support currently only in OPAL-T.
-restart	-1 – <Integer>	restarts from given step in file with saved phase space. Per default OPAL tries to restart from a file <file>.h5 where <file> is the input file without extension. -1 stands for the last step in the file. If no other file is specified to restart from and if the last step of the file is chosen, then the new data is appended to the file. Otherwise the data from this particular step is copied to a new file and all new data appended to the new file.
-restartfn	<file>	a file in H5hut format from which OPAL should restart.

Example:

```
opal input.in -restartfn input.h5 -restart -1 --info 3
```

3.2 Autophase Example

This is a partial complete example. First we have to set OPAL in AUTOPHASE mode, as described in Section 7.3 and for example set the nominal phase to -3.5 (deg). The way how OPAL is computing the phases is explained in Appendix E.

```
Option, AUTOPHASE=4;

FINSS_RGUN_phi= (-3.5/180*Pi);
```

The cavity would be defined like

```
FINSS_RGUN: RFCavity, L = 0.17493, VOLT = 100.0,
    FMAPFN = "FINSS-RGUN.dat",
    ELEMEDGE =0.0, TYPE = "STANDING", FREQ = 2998.0,
    LAG = FINSS_RGUN_phi;
```

with FINSS_RGUN_phi defining the off crest phase. Now a normal TRACK command can be executed. A file containing the values of maximum phases is created, and has the format like:

```
1
FINSS_RGUN
2.22793
```

with the first entry defining the number of cavities in the simulation.

3.3 Examples of Particle Accelerators and Beamlines

```
Title,string="OBLA Gun";      examples/OBLA-Gun/OBLA-Gun.in
```

```

Option, ECHO=FALSE;
Option, PSDUMPFREQ=10;

Edes=1.0E-9;
gamma=(Edes+EMASS)/EMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*EMASS;
brho = (EMASS*1.0e9*gambet) / CLIGHT;

value,{gamma,brho,Edes,beta,gambet};

// L:      physical element length (real)
// KS:      field scaling factor (real)
// FMAPFN:  field file name (string)
// ELEMEDGE: physical start of the element on the floor (real)
SP1: Solenoid, L=1.20, ELEMEDGE=-0.5335, FMAPFN="1T2.T7",
      KS=0.00011;
SP2: Solenoid, L=1.20, ELEMEDGE=-0.399, FMAPFN="1T3.T7", KS=0.0;
SP3: Solenoid, L=1.20, ELEMEDGE=-0.269, FMAPFN="1T3.T7", KS=0.0;

gun: RFCavity, L=0.011, VOLT=-102.28, FMAPFN="1T1.T7",
      ELEMEDGE=0.00, TYPE="STANDING", FREQ=1.0e-6;

l1:   Line = (gun,sp1,sp2,sp3);

rf=1498.956e6;
v0=beta*CLIGHT;
lz = 6.5E-12*v0;
value,{v0,lz};

Dist1:DISTRIBUTION, DISTRIBUTION=gungauss,
      sigma= 0.00030, sigmapx=0.0, corrx=0.0,
      sigmay= 0.00030, sigmapy=0.0, corry=0.0,
      sigmat= lz, sigmapt=1.0, corrt=0.0 ,
      TEMISSION=3.9e-11, NBIN=39, DEBIN=1;

MINSTEPFORREBIN=1000;

Fsl:FIELDSOLVER, FSTYPE=FFT, MX=32, MY=32, MT=32,
      PARFFTX=true, PARFFTY=true, PARFFT=false,
      BCFFTX=open, BCFFTY=open, BCFFT=open,
      BBOXINCR=1, GREENSF=STANDARD;

beam1: BEAM, PARTICLE=ELECTRON, pc=P0, NPART=50000,
      BCURRENT=0.008993736, BFREQ=rf, CHARGE=-1;

Select, Line=l1;
track,line=l1, beam=beam1, MAXSTEPS=2000, DT=1.0e-13;
run, method = "PARALLEL-T", beam=beam1, fieldsolver=Fsl,
      distribution=Dist1;
endtrack;
Stop;

```

3.3.1 PSI XFEL 250 MeV Injector

```

examples/diagnostic.in
Option, ECHO=FALSE;
Option, TFS=FALSE;
Option, PSDUMPFREQ=10;

Title,string="OPAL Diagnostics test";

Edes=0.0307; //GeV
gamma=(Edes+EMASS)/EMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*EMASS;
brho = (EMASS*1.0e9*gambet) / CLIGHT;
value,{gamma,brho,Edes,beta,gambet};

FINLB02_MSLAC40: Solenoid, L=0.001, KS=0.05,
                  FMAPFN="FINLB02-MSLAC.T7", ELEMEDGE=4.554;

FIND1_MQ10: Quadrupole, L=0.1, K1=2.788, ELEMEDGE=5.874;
FIND1_MQ20: Quadrupole, L=0.1, K1=-3.517, ELEMEDGE=6.074;

SCREEN: Monitor, L=0.01, ELEMEDGE=7.3867, OUTFN="Screen.h5";

FIND1: Line = (FINLB02_MSLAC40, FIND1_MQ10, FIND1_MQ20, SCREEN);

Dist1:DISTRIBUTION, DISTRIBUTION=gauss,
      sigmax= 1.0e-03, sigmapx=1.0e-4, corrx=0.5,
      sigmay= 2.0e-03, sigmapy=1.0e-4, corry=-0.5,
      sigmat= 3.0e-03, sigmapt=1.0e-4, corrt=0.0;

Fs2:FIELDSOLVER, FSTYPE=FFT, MX=32, MY=32, MT=64,
     PARFFTX=false, PARFFTY=false, PARFFT=true,
     BCFFTX=open, BCFFTY=open, BCFFT=open,
     BBOXINCR=1.0, GREENSF=INTEGRATED;

beam1: BEAM, PARTICLE=ELECTRON, pc=P0, NPART=1e5,
       BFREQ=1498.953425154e6, BCURRENT=0.299598, CHARGE=-1;

Select, Line=FIND1;

track,line=FIND1, beam=beam1, MAXSTEPS=10000, DT=1.0e-12;
run, method = "PARALLEL-T", beam=beam1, fieldsolver=Fs2,
  distribution=Dist1;
endtrack;
Stop;

```

3.3.2 PSI Injector II Cyclotron

Injector II is a separated sector cyclotron specially designed for preacceleration (inject: 870 keV, extract: 72 MeV) of high intensity proton beam for Ring cyclotron. It has 4 sector magnets, two double-gap acceleration cavities (represented by 2 single-gap cavities here) and two single-gap flat-top cavities.

Following is an input file of **Single Particle Tracking mode** for PSI Injector II cyclotron.

```

examples/opal-cycl.in
Title,string="OPAL-CyclT test";

Option, ECHO=FALSE;
Option, PSDUMPFREQ=100000;
Option, SPTDUMPFREQ=5;

```

```

// define some physical parameters
Edes=0.002;
gamma=(Edes+PMASS)/PMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*PMASS;
brho = (PMASS*1.0e9*gambet) / CLIGHT;
value,{gamma,brho,Edes,beta,gambet};
phi01= 48.4812-15.0;
phi02=phi01+200.0;
phi03=phi01+1800.0;
phi04=phi01+2000.0;
phi05=(phi01+820.0)*3.0;
phi06=(phi01+2620.0)*3.0;

turns = 106;
nstep=5000;

// define elements and beamline
inj2: Cyclotron, TYPE="Injector2", CYHARMON=10, PHIINIT=30.0,
    PRINIT=-0.0067, RINIT=392.5, SYMMETRY=1.0,
    RFFREQ=frequency, FMAPFN="..ZYKL9Z.NAR";

rf0: RFCavity, VOLT=0.25796, FMAPFN="..av1.dat",
    TYPE="SINGLE GAP", FREQ=frequency, ANGLE=35.0, PHI0=phi01,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf1: RFCavity, VOLT=0.25796, FMAPFN="..av1.dat",
    TYPE="SINGLE GAP", FREQ=frequency, ANGLE=55.0, PHI0=phi02,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf2: RFCavity, VOLT=0.25796, FMAPFN="..av1.dat",
    TYPE="SINGLE GAP", FREQ=frequency, ANGLE=215.0, PHI0=phi03,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf3: RFCavity, VOLT=0.25796, FMAPFN="..av1.dat",
    TYPE="SINGLE GAP", FREQ=frequency, ANGLE=235.0, PHI0=phi04,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf4: RFCavity, VOLT=0.06380, FMAPFN="..av3.dat",
    TYPE="SINGLE GAP", FREQ=frequency3, ANGLE=135.0, PHI0=phi05,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf5: RFCavity, VOLT=0.06380, FMAPFN="..av3.dat",
    TYPE="SINGLE GAP", FREQ=frequency3, ANGLE=315.0, PHI0=phi06,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

l1: Line = (inj2,rf0,rf1,rf2,rf3,rf4,rf5);

// define particles distribution, read from file
Dist1:DISTRIBUTION, DISTRIBUTION=fromfile,
    FNAME="PartDatabase.dat";

// usset define fieldsolver, useless for single particle track mode
Fs1:FIELDSOLVER, FSTYPE=NONE, MX=64, MY=64, MT=64,
    PARFFTX=true, PARFFTY=true, PARFFTT=false,
    BCFFTX=open, BCFFTY=open, BCFFTT=open;

// define beam parameters
beam1: BEAM, PARTICLE=PROTON, pc=P0, NPART=1, BCURRENT=0.0;

// select beamline
Select, Line=l1;

```

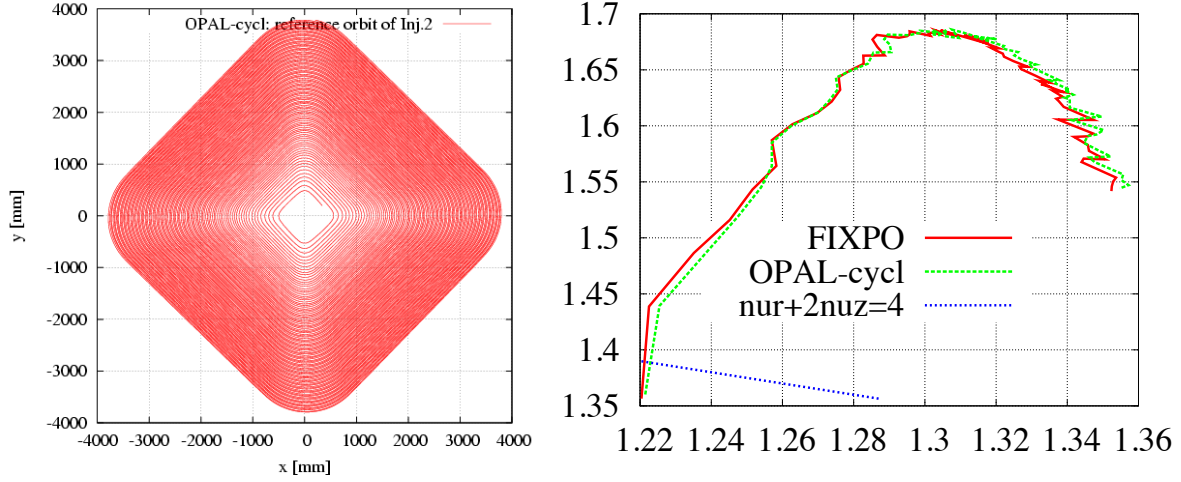


Figure 3.1: Reference orbit(left) and tune diagram(right) in Injector II

```
// start tracking
track,line=11, beam=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep,TIMEINTEGRATOR="RK-4";
run, method = "CYCLOTRON-T", beam=beam1, fieldsolver=Fsl, distribution=Dist1;
endtrack;
Stop;
```

To run opal on a single node, just use this command:

```
opal testinj2-1.in
```

Here shows some pictures using the resulting data from single particle tracking using OPAL-CYCL.

Left plot of Figure 3.1 shows the accelerating orbit of reference particle. After 106 turns, the energy increases from 870 keV at the injection point to 72.16 MeV at the deflection point.

From theoretic view, there should be an eigen ellipse for any given energy in stable area of a fixed accelerator structure. Only when the initial phase space shape matches its eigen ellipse, the oscillation of beam envelop amplitude will get minimal and the transmission efficiency get maximal. We can calculate the eigen ellipse by single particle tracking using betatron oscillation property of off-centered particle as following: track an off-centered particle and record its coordinates and momenta at the same azimuthal position for each revolution. Figure 3.2 shows the eigen ellipse at symmetric line of sector magnet for energy of 2 MeV in Injector II.

Right plot of Figure 3.1 shows very good agreement of the tune diagram by OPAL-CYCL and FIXPO. The trivial discrepancy should come from the methods they used. In FIXPO, the tune values are obtained according to the crossing points of the initially displaced particle. Meanwhile, in OPAL-CYCL, the Fourier analysis method is used to manipulate orbit difference between the reference particle and an initially displaced particle. The frequency point with the biggest amplitude is the betatron tune value at the given energy.

Following is the input file for single bunch tracking with space charge effects in Injector II.

```
// file name ''testinj2-2.in'' examples/opal-cycl-sc.in
// Define Option parameters
Option, ECHO=FALSE;
Option, PSDUMPFREQ=10;
Option, SPTDUMPFREQ=5;
Option, REPARTFREQ=10;
```

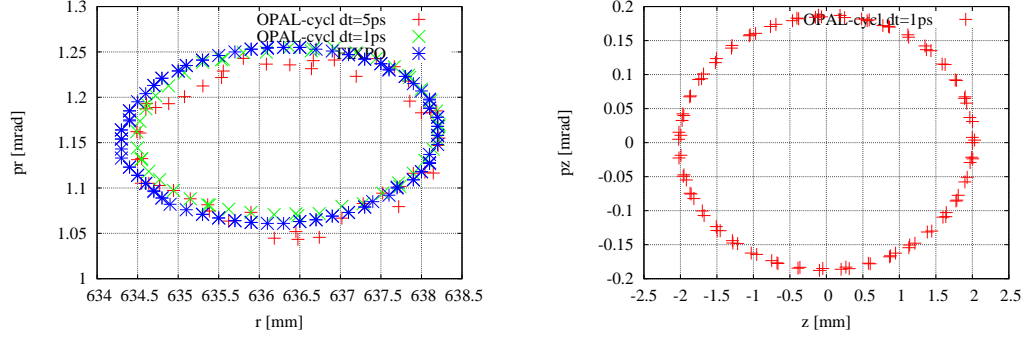


Figure 3.2: Radial and vertical eigenellipse at 2 MeV of Injector II

```
// Define title
Title,string="OPAL-CyclT-SC test";

// define some physical parameters
Edes=0.000870;
gamma=(Edes+PMASS)/PMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*PMASS;
brho = (PMASS*1.0e9*gambet) / CLIGHT;

phi01= 48.4812-15.0;
phi02=phi01+200.0;
phi03=phi01+1800.0;
phi04=phi01+2000.0;
phi05=(phi01+820.0)*3.0;
phi06=(phi01+2620.0)*3.0;

frequency=50.6370;
frequency3=3.0*frequency;

turns = 106;
nstep=5000;

// define elements and beamline
inj2: Cyclotron, TYPE="Injector2", CYHARMON=10, PHIINIT=30.0,
    PRINIT=-0.0067, RINIT=392.5, SYMMETRY=1.0,
    RFFREQ=50.6370, FMAPFN=".../ZYKL9Z.NAR";

rf0: RFCavity, VOLT=0.25796, FMAPFN=".../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=35.0, PHI0=phi01,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf1: RFCavity, VOLT=0.25796, FMAPFN=".../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=55.0, PHI0=phi02,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf2: RFCavity, VOLT=0.25796, FMAPFN=".../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=215.0, PHI0=phi03,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf3: RFCavity, VOLT=0.25796, FMAPFN=".../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=235.0, PHI0=phi04,
```

```

RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf4: RFCavity, VOLT=0.06380, FMAPFN="../../Cav3.dat",
    TYPE="SINGLE GAP", FREQ=151.911, ANGLE=135.0, PHI0=phi05,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf5: RFCavity, VOLT=0.06380, FMAPFN="../../Cav3.dat",
    TYPE="SINGLE GAP", FREQ=151.911, ANGLE=315.0, PHI0=phi06,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

11: Line = (inj2,rf0,rf1,rf2,rf3,rf4,rf5);

// define particles distribution, generate Gaussian type
Dist1:DISTRIBUTION, DISTRIBUTION=gauss,
    sigmax= 2.0e-03, sigmapx=1.0e-7, corrx=0.0,
    sigmay= 2.0e-03, sigmapy=1.0e-7, corry=0.0,
    sigmat= 2.0e-03, sigmapt=1.8e-4, corrt=0.0;

// define parallel FFT fieldsolver, parallel in x y direction
Fs1:FIELDSOLVER, FSTYPE=FFT, MX=32, MY=32, MT=32,
    PARFFTX=true, PARFFTY=true, PARFFT=false,
    BCFFTX=open, BCFFTY=open, BCFFT=open, BBOXINCR=5;

// define beam parameters
beam1: BEAM, PARTICLE=PROTON, pc=P0,
    NPART=1e5, BCURRENT=3.0E-3, CHARGE=1.0, BFREQ= frequency;

// select beamline
Select, Line=11;

// start tracking
track,line=11, beam=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep, TIMEINTEGRATOR="LF-2";
run, method = "CYCLOTRON-T", beam=beam1, fieldsolver=Fs1, distribution=Dist1;
endtrack;
Stop;

```

To run opal on single node, just use this command:

```
# opal testinj2-2.in
```

To run opal on N nodes in parallel environment interactively, use this command instead:

```
# mpirun -np N opal testinj2-2.in
```

If restart a job from the last step of an existing .h5 file, add a new argument like this:

```
# mpirun -np N opal testinj2-2.in -restart -1
```

Figure 3.3 and 3.4 are simulation results, shown by H5ROOT code.

3.3.3 PSI Ring Cyclotron

From the view of numerical simulation, the difference between Injector II and Ring cyclotron comes from two aspects:

B Field The structure of Ring is totally symmetric, the field on median plane is periodic along azimuthal direction, OPAL-CYCL take this advantage to only store 1/8 field data to save memory.

RF Cavity In the Ring, all the cavities are typically single gap with some parallel displacement from its radial position. OPAL-CYCL have an argument PDIS to manipulate this issue.

Figure 3.5 shows a single particle tracking result and tune calculation result in the PSI Ring cyclotron. Limited by size of the user guide, we don't plan to show too much details as in Injector II.

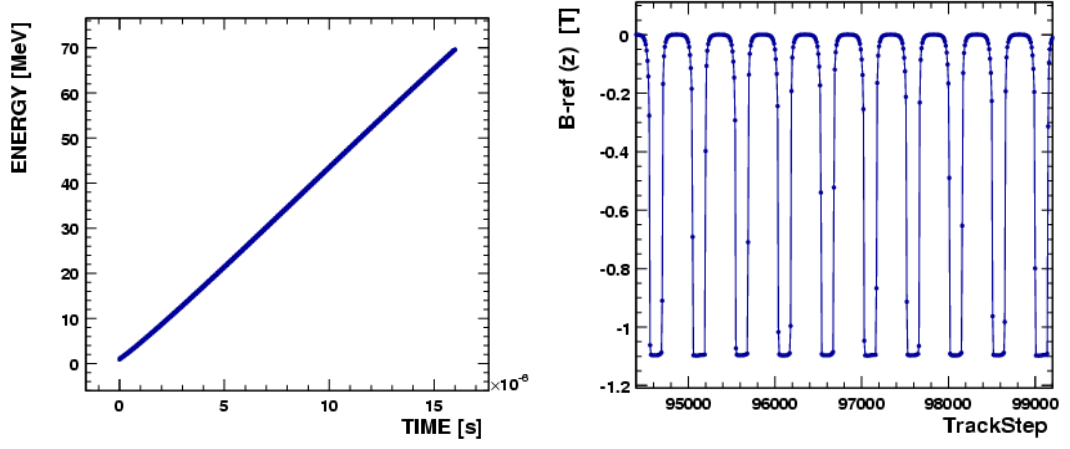


Figure 3.3: Energy Vs. time (left) and external B field Vs. trackstep (Right, only show for about 2 turns)

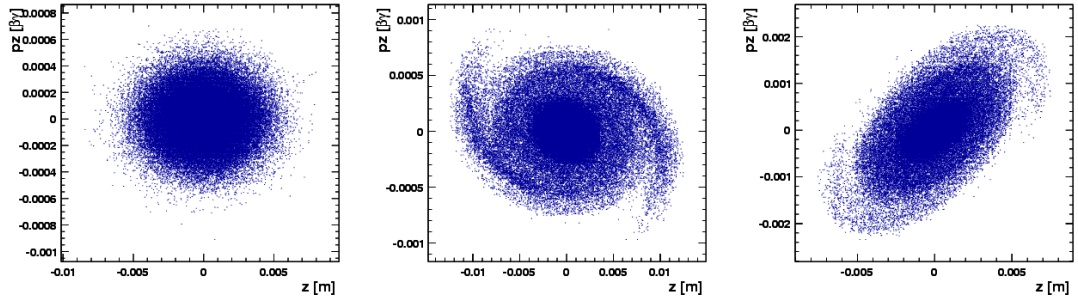


Figure 3.4: Vertical phase at different energy from left to right: 0.87 MeV, 15 MeV and 35 MeV

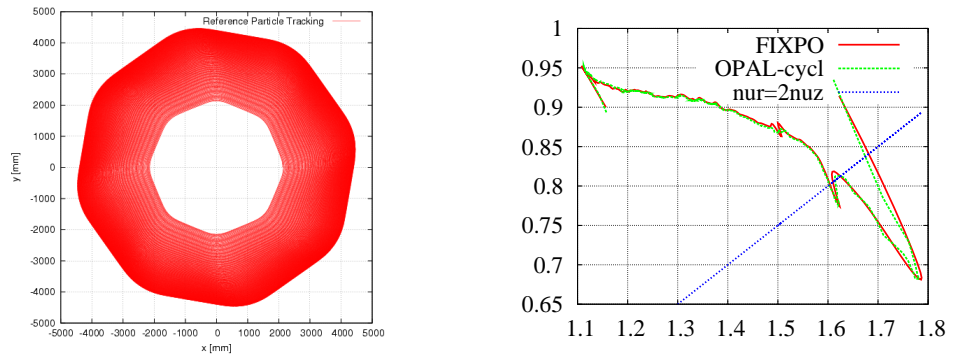


Figure 3.5: Reference orbit(left) and tune diagram(right) in Ring cyclotron

3.4 Translate Old Distribution Commands to New Distribution Commands

As of OPAL 1.2, the distribution command (Chapter 11) was changed significantly. Many of the changes were internal to the code, allowing us to more easily add new distribution command options. However, other changes were made to make creating a distribution easier, clearer and so that the command attributes were more consistent across distribution types. Therefore, we encourage our users to refer to Chapter 11 when creating any new input files, or if they wish to update existing input files.

With the new distribution command, we did attempt as much as possible to make it backward compatible so that existing OPAL input files would still work the same as before, or with small modifications. In this section of the manual, we will give several examples of distribution commands that will still work as before, even though they have antiquated command attributes. We will also provide examples of commonly used distribution commands that need small modifications to work as they did before.

An important point to note is that it is very likely you will see small changes in your simulation even when the new distribution command is nominally generating particles in exactly the same way. This is because random number generators and their seeds will likely not be the same as before. These changes are only due to OPAL using a different sequence of numbers to create your distribution, and not because of errors in the calculation. (Or at least we hope not.)

3.4.1 GUNGAUSSFLATTOPTH and ASTRAFLATTOPTH Distribution Types

The GUNGAUSSFLATTOPTH and ASTRAFLATTOPTH distribution types (See Sections 11.5.4 and 11.5.5) are two common types previously implemented to simulate electron beams emitted from photocathodes in an electron photoinjector. These are no longer explicitly supported and are instead now defined as specialized sub-types of the distribution type FLATTOP. That is, the *emitted* distributions represented by GUNGAUSSFLATTOPTH and ASTRAFLATTOPTH can now be easily reproduced by using the FLATTOP distribution type and we would encourage use of the new command structure.

Having said this, however, old input files that use the GUNGAUSSFLATTOPTH and ASTRAFLATTOPTH distribution types will still work as before, with the following exception. Previously, OPAL had a boolean OPTION command FINEEMISSION (default value was TRUE). This OPTION is no longer supported. Instead you will need to set the distribution attribute EMISSIONSTEPS (Table 11.3) to a value that is $10 \times$ the value of the distribution attribute NBIN (Table 11.4) in order for your simulation to behave the same as before.

3.4.2 FROMFILE, GAUSS and BINOMIAL Distribution Types

The FROMFILE (Section 11.3), GAUSS (Section 11.4) and BINOMIAL (Section 11.6) distribution types have changed from previous versions of OPAL. However, legacy distribution commands should work as before with one exception. If you are using OPAL-CYCL then your old input files will work just fine. However, if you are using OPAL-T then you will need to set the distribution attribute INPUTMOUNITS (Section 11.1) to:

INPUTMOUNITS = EV

Chapter 4

OPAL-T

4.1 Introduction

OPAL-T is a fully three-dimensional program to track in time, relativistic particles taking into account space charge forces, self-consistently in the electrostatic approximation, and short-range longitudinal and transverse wake fields. OPAL-T is one of the few codes that is implemented using a parallel programming paradigm from the ground up. This makes OPAL-T indispensable for high statistics simulations of various kinds of existing and new accelerators. It has a comprehensive set of beamline elements, and furthermore allows arbitrary overlap of their fields, which gives OPAL-T a capability to model both the standing wave structure and traveling wave structure. Beside IMPACT-T it is the only code making use of space-charge solvers based on an integrated Green [39, 40, 41] function to efficiently and accurately treat beams with large aspect ratio, and a shifted Green function to efficiently treat image charge effects of a cathode [38, 39, 40, 41]. For simulations of particles sources i.e. electron guns OPAL-T uses the technique of energy binning in the electrostatic space-charge calculation to model beams with large energy spread. In the very near future a parallel Multi Grid solver taking into account the exact geometry will be implemented.

4.2 Variables in OPAL-T

OPAL-T uses the following canonical variables to describe the motion of particles. The physical units are listed in square brackets.

X Horizontal position x of a particle relative to the axis of the element [m].

PX $\beta_x \gamma$ Horizontal canonical momentum [1].

Y Horizontal position y of a particle relative to the axis of the element [m].

PY $\beta_y \gamma$ Horizontal canonical momentum [1].

Z Longitudinal position z of a particle in floor co-ordinates [m].

PZ $\beta_z \gamma$ Longitudinal canonical momentum [1].

The independent variable is **t** [s].

4.3 Integration of the Equation of Motion

OPAL-T integrates the relativistic Lorentz equation

$$\frac{d\gamma\mathbf{v}}{dt} = \frac{q}{m} [\mathbf{E}_{\text{ext}} + \mathbf{E}_{\text{sc}} + \mathbf{v} \times (\mathbf{B}_{\text{ext}} + \mathbf{B}_{\text{sc}})] \quad (4.1)$$

where γ is the relativistic factor, q is the charge, and m is the rest mass of the particle. \mathbf{E} and \mathbf{B} are abbreviations for the electric field $\mathbf{E}(\mathbf{x}, t)$ and magnetic field $\mathbf{B}(\mathbf{x}, t)$.

4.4 Envelope Tracker

The OPAL-e Envelope Tracker is an algorithm used to solve the envelope equations of a beam propagating through external fields and under the influence of its own space charge. The algorithm is based on the multi-slice analysis approach used in the theory of emittance compensation [63]. The space-charge model used can be switched between an analytic model derived and used in HOMDYN [64] and a similar model developed at PSI called Beam Envelope Tracker (BET).

4.4.1 Envelope Equation without Dispersion For Long Beams

The equation for the propagation of a general beam enveloped given here follows the work of Sacherer [72]. Using the variables x and p_x as the phase space variables, the equation of motion for $\sigma_x = \langle x^2 \rangle^{1/2}$ is found by differentiating with respect to time:

$$\dot{\sigma}_x = \frac{\langle x\dot{x} \rangle}{\sigma}, \quad \ddot{\sigma}_x = \frac{1}{\sigma_x^3} [\langle x^2 \rangle \langle \dot{x}^2 \rangle - \langle x\dot{x} \rangle^2] + \frac{\langle x\ddot{x} \rangle}{\sigma_x} \quad (4.2)$$

Noting that $\dot{x} = p_x/m\gamma$, the above equations become

$$\ddot{\sigma}_x = \frac{1}{\sigma_x^3} \left(\frac{c\varepsilon_n}{\gamma} \right)^2 + \frac{\langle x\ddot{x} \rangle}{\sigma_x}, \quad (4.3)$$

where the normalized emittance is defined as $\varepsilon_{n,x} = \frac{1}{mc} \sqrt{\langle x^2 \rangle \langle p_x^2 \rangle - \langle xp_x \rangle^2}$. The last term in this equation is expanded using $\ddot{x} = -\gamma^2 \beta \dot{\beta} \dot{x} + F_x/m\gamma$:

$$\ddot{\sigma}_x = [-\gamma^2 \beta \dot{\beta}] \dot{\sigma}_x + \frac{\langle xF_x \rangle}{m\gamma\sigma_x} + \left(\frac{c\varepsilon_n}{\gamma} \right)^2 \frac{1}{\sigma_x^3}, \quad (4.4)$$

The force is split into a (linear) external part and the self-fields: $F_x(t, x) = -K_x(t)x + F_{x,s}$. Plugging this into the envelope equation gives

$$\ddot{\sigma}_x + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_x + \left[\frac{K}{m\gamma} \right] \sigma_x = \left[\frac{\langle xF_{x,s} \rangle}{m\gamma} \right] \frac{1}{\sigma_x} + \left(\frac{c\varepsilon_n}{\gamma} \right)^2 \frac{1}{\sigma_x^3}. \quad (4.5)$$

Following Sacherer [72], the term $\langle xF_{x,s} \rangle$ can be interpreted as the linear part of the space charge field (in a least squares sense). The linear part of the field is defined by $F_{x,s}^{(1)}$, such that the quantity

$$D = \langle (F_{x,s}^{(1)}x - F_{x,s})^2 \rangle = \int (F_{x,s}^{(1)}x - F_{x,s})^2 \rho_x dx, \quad (4.6)$$

is minimized in a least squares sense. This is accomplished by setting $\partial D / \partial F_{x,s}^{(1)} = 0$, which implies [72]:

$$F_{x,s}^{(1)}x = \frac{\langle xF_{x,s} \rangle}{\sigma_x^2} x. \quad (4.7)$$

This gives the general form of the envelope equation

$$\begin{aligned}\ddot{\sigma}_x + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_x + \left[\frac{K}{m\gamma} \right] \sigma_x &= \left[\frac{F_{x,s}^{(1)}}{m\gamma} \right] \sigma_x + \left(\frac{c\varepsilon_n}{\gamma} \right)^2 \frac{1}{\sigma_x^3}, \\ \sigma_x'' + \left[\frac{\gamma'}{\beta^2 \gamma} \right] \sigma_x' + \left[\frac{K}{mc^2 \beta p_n} \right] \sigma_x &= \left[\frac{F_{x,s}^{(1)}}{mc^2 \beta p_n} \right] \sigma_x + \left(\frac{\varepsilon_n}{p_n} \right)^2 \frac{1}{\sigma_x^3}.\end{aligned}\quad (4.8)$$

In this equation $p_n = \beta\gamma$ is the normalized momentum.

A general form for the fields can be now introduced. For a long beam, the linear part of the fields (in the beam rest frame) are given by

$$E_x = \left(\frac{\partial E_x}{\partial x} \Big|_0 \right) x, \quad E_y = \left(\frac{\partial E_y}{\partial y} \Big|_0 \right) y. \quad (4.9)$$

These fields must then be boosted back to the laboratory frame according to

$$\begin{aligned}\mathbf{E} &= \gamma(\mathbf{E}' - \beta \times \mathbf{cB}') - \frac{\gamma^2}{1+\gamma} \beta(\beta \cdot \mathbf{E}'), \\ \mathbf{B} &= \gamma(\mathbf{B}' + \beta \times \mathbf{E}'/c) - \frac{\gamma^2}{1+\gamma} \beta(\beta \cdot \mathbf{B}'),\end{aligned}\quad (4.10)$$

For a beam moving along the z -axis with speed $c\beta$, the fields are given by

$$\mathbf{E} = \gamma E'_x \hat{\mathbf{x}} + \gamma E'_y \hat{\mathbf{y}}, \quad \mathbf{B} = \frac{\gamma\beta}{c} (-E'_y \hat{\mathbf{x}} + E'_x \hat{\mathbf{y}}). \quad (4.11)$$

Here the primes label the field components in the lab frame. Assuming these components are proportional to the line charge density λ' the fields can be written in a form that takes into account the Lorentz contraction of the current density:

$$E_x = \gamma E'_x = \gamma \lambda' \frac{\partial E'_x}{\partial \lambda'} = \lambda \frac{\partial E'_x}{\partial \lambda'}. \quad (4.12)$$

The linearized force equation then takes the form

$$\mathbf{F} = e(E_x - \beta c B_y) \hat{\mathbf{x}} + e(E_y + \beta c B_x) \hat{\mathbf{y}} = \frac{e\lambda}{\gamma^2} \left[\left(\frac{\partial^2 E'_x}{\partial x \partial \lambda'} \right) x \hat{\mathbf{x}} + \left(\frac{\partial^2 E'_y}{\partial y \partial \lambda'} \right) y \hat{\mathbf{y}} \right].$$

With this, the envelope equations become

$$\ddot{\sigma}_i + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_i + \left[\frac{K_i}{m\gamma} \right] \sigma_i = \left[\frac{e\lambda}{m\gamma^3} \left(\frac{\partial^2 E'_i}{\partial x_i \partial \lambda'} \right) \right] \sigma_i + \left(\frac{c\varepsilon_{n,i}}{\gamma} \right)^2 \frac{1}{\sigma_i^3}. \quad (4.13)$$

So far this derivation has assumed that there is no coupling between x and y in both the external and self forces. In the presence of solenoids this is no longer the case for the external fields. If the beam and external fields are cylindrically symmetric then the previous analysis can be performed in the Larmor frame. Working in the Larmor frame, the equations of motion for $\sigma_x = \sigma_y = \sigma_L$ decouple and the envelope equation is given by

$$\ddot{\sigma}_L + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_L + \left[\frac{K}{m\gamma} + (\dot{\theta}_r)^2 \right] \sigma_L = \left[\frac{e\lambda}{m\gamma^3} \left(\frac{\partial^2 E'_r}{\partial r \partial \lambda'} \right) \right] \sigma_L + \left(\frac{c\varepsilon_n}{\gamma} \right)^2 \frac{1}{\sigma_L^3}.$$

In this expression, θ_r is the Larmor angle, and is given by

$$\theta_r = - \int \left(\frac{eB_z}{2m\gamma} \right) dt = - \int \left(\frac{eB_z}{2m\gamma\beta c} \right) dz. \quad (4.14)$$

Long Uniform Cylindrical Beam

The envelope equation for a cylindrical beam is now explicitly derived. Assuming cylindrically symmetric fields and working in the Larmor frame then $\sigma_x = \sigma_y = \sigma_L$. It is important to distinguish this parameter from $\sigma_r = R/\sqrt{2}$. In fact, $\sigma_L = R/2$ for a circular beam. The electric in the lab frame can be easily computed from Gauss's law:

$$E'_r = \frac{1}{8\pi\epsilon_0} \frac{\lambda'}{\sigma^2} r \quad (r < R). \quad (4.15)$$

This is the only portion of the field needed since the charge is zero outside of the cylinder radius, and the average $\langle xF_x \rangle$ is zero there. The space charge term is then given by

$$\frac{e\lambda}{m\gamma^3} \left(\frac{\partial^2 E'_r}{\partial r \partial \lambda'} \right) = \frac{c^2}{2\beta\sigma^2} \left(\frac{e}{4\pi\epsilon_0 mc^3} \right) \lambda\beta c = \frac{c^2}{\beta\sigma^2} \frac{I}{2I_0}, \quad (4.16)$$

where the Alfven current is identified as the factors in parenthesis. With this the envelope equation is written in its 'standard' form [63]

$$\ddot{\sigma}_L + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_L + \left[\frac{K}{m\gamma} + (\dot{\theta}_r)^2 \right] \sigma_L = \left[\frac{c^2 k_p}{\beta\gamma^3} \right] \frac{1}{\sigma_L} + \left(\frac{c\epsilon_n}{\gamma} \right)^2 \frac{1}{\sigma_L^3},$$

where the term $k_p = I/2I_0$ is the beam perveance. Equivalently, these equations can be written in terms of the beam radius R :

$$\ddot{R} + [\gamma^2 \beta \dot{\beta}] \dot{R} + \left[\frac{K}{m\gamma} + (\dot{\theta}_r)^2 \right] R = \left[\frac{4c^2 k_p}{\beta\gamma^3} \right] \frac{1}{R} + \left(\frac{4c\epsilon_n}{\gamma} \right)^2 \frac{1}{R^3}. \quad (4.17)$$

Long Uniform Elliptical Beam

Here an intuitive argument is used to find the linear portions of the fields from a uniform elliptical beam. The beam's cross-section is defined so that the ellipse axes line up with the x and y cartesian coordinates and are given by X and Y respectively. The symmetry of the problem in cartesian coordinates implies the following constraints on the transverse electric field components in the beam rest frame:

$$\begin{cases} E_x(-x, y, z) = -E_x(x, y, z) \\ E_x(x, -y, z) = E_x(x, y, z) \end{cases}, \quad \begin{cases} E_y(-x, y, z) = E_y(x, y, z) \\ E_y(x, -y, z) = -E_y(x, y, z) \end{cases}, \quad (4.18)$$

These conditions imply that field components can be expanded as

$$E_x = \sum_{n,m=0}^{\infty} E_x^{(2n+1,2m)} x^{2n+1} y^{2m}, \quad E_y = \sum_{n,m=0}^{\infty} E_y^{(2n,2m+1)} x^{2n} y^{2m+1}. \quad (4.19)$$

Similarly the charge density must satisfy $\rho(-x, -y) = \rho(x, y)$, implying:

$$\rho(x, y) = \sum_{n,m=0}^{\infty} \rho^{(2n,2m)} x^{2n} y^{2m}. \quad (4.20)$$

Gauss's law can then be use to relate the expansion coefficients:

$$(2n+1)E_x^{(2n+1,2m)} + (2m+1)E_y^{(2n,2m+1)} = \frac{1}{\epsilon_0} \rho^{(2n,2m)} \quad (4.21)$$

The linear components of the field (inside the beam) are given by taking $n = m = 0$. This implies that

$$\left. \frac{\partial E_x}{\partial x} \right|_{r=0} + \left. \frac{\partial E_y}{\partial y} \right|_{r=0} = \frac{\lambda}{\pi\epsilon_0 XY}. \quad (4.22)$$

It is now assumed that for the linear portion of the fields, the above derivatives have a similar form. The assumed form is

$$\left. \frac{\partial E_x}{\partial x} \right|_{r=0} = \frac{f}{X}, \quad \left. \frac{\partial E_y}{\partial y} \right|_{r=0} = \frac{f}{Y}. \quad (4.23)$$

The form here is chosen so that the field components have the right behavior in the limit that X or Y goes to zero, as well as the limit $X = Y = R$. With this assumption, the linear field components can be solved for:

$$E_x = \frac{\lambda x}{\pi \varepsilon_0 X(X+Y)}, \quad E_y = \frac{\lambda y}{\pi \varepsilon_0 Y(X+Y)}. \quad (4.24)$$

For a more rigorous derivation of this result, please see [72]. From these expressions the envelope equations can be derived using the Eq. (4.13):

$$\ddot{\sigma}_i + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_i + \left[\frac{K_i}{m\gamma} \right] \sigma_i = \left[\frac{2c^2}{\beta \gamma^3} \right] \frac{k_p}{\sigma_x + \sigma_y} + \left(\frac{c\varepsilon_{n,i}}{\gamma} \right)^2 \frac{1}{\sigma_i^3}. \quad (4.25)$$

The equivalent equations for the ellipse axes are the given by

$$\ddot{X}_i + [\gamma^2 \beta \dot{\beta}] \dot{X}_i + \left[\frac{K_i}{m\gamma} \right] X_i = \left[\frac{8c^2}{\beta \gamma^3} \right] \frac{k_p}{X+Y} + \left(\frac{4c\varepsilon_{n,i}}{\gamma} \right)^2 \frac{1}{X_i^3}. \quad (4.26)$$

4.4.2 Slice Model

The slice model deployed in OPAL-e is based on the slice analysis of the envelope equation developed in emittance compensation theory [63]. In this analysis the beam is taken to be laminar in both the transverse and longitudinal directions. The beam is then cut into slices along the length of the bunch. The slices are taken to be independent, and the transverse envelope equation applied to each. As a result, each slice changes both in length and radius as

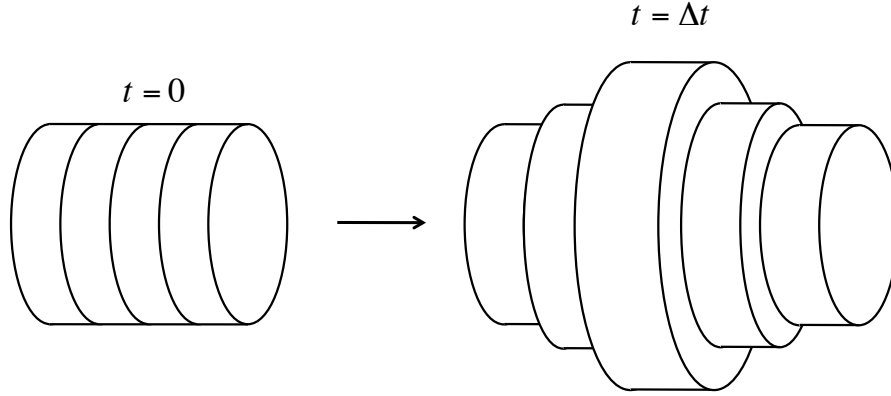


Figure 4.1: Evolution of the particle beam in the multi-slice approximation.

the beam propagates through the accelerator. This is shown schematically in Fig. 4.1. The position of each slice is labeled by the coordinate ζ . The beam envelope of each slice is then written as $\sigma = \sigma(t, z, \zeta)$. Additionally, for very long beams, the current is given by

$$I = I(t, z, \zeta) = c\beta(\zeta)\lambda(\zeta). \quad (4.27)$$

In the case of bunched beams, the current picks up an extra factor:

$$I \rightarrow I(\zeta)g(\zeta). \quad (4.28)$$

The term $g(\zeta)$ is determined by the bunch distribution, and is less than or equal to unity. It is noted that in this model the slices are assumed to have some infinitesimal width $\delta\zeta = \delta\zeta(t, z)$ which is the same for all slices. For a bunched beam with length L this is given by $\delta\zeta = L/N$, where N is the number of slices. The position of each slice evolves according

$$\dot{\beta}(t, \zeta) = \frac{eE_z}{mc\gamma^3(\zeta)}. \quad (4.29)$$

The longitudinal field here includes both the external and self-fields (for bunched beams) at the slice position.

The HOMDYN Model: Analytic Space Charge Fields for Bunched Beams

In this section the analytic formula for the linear Electric fields from a uniformly charged cylinder are derived. This done in two steps. First the on-axis field from a uniformly charged circular disk is calculated. The infinitesimal field contribution in this case is

$$d\mathbf{E} = \frac{1}{4\pi\epsilon_0} \frac{(\mathbf{x} - \mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|^3} \sigma dA',$$

with $\mathbf{x} - \mathbf{x}' = (z - z')\hat{\mathbf{z}} - r'\hat{\mathbf{r}}'$ on-axis. Noting that the integration over the radial component vanishes on-axis, the field from the disk can be readily integrated, yielding:

$$\mathbf{E}_z^{\text{disk}}(r=0) = \frac{\sigma}{2\epsilon_0} \left(\frac{(z - z')}{|z - z'|} - \frac{(z - z')}{\sqrt{(z - z')^2 + R^2}} \right) \hat{\mathbf{z}}. \quad (4.30)$$

From this, the on-axis field for the cylinder can be found by letting $\sigma = \rho_0 dz'$ and integrating up the contribution from infinitesimal charged disks from $0 \leq z' \leq L$:

$$E_z(=0) = \frac{Q}{2\pi\epsilon_0 R^2} H(z, A), \quad (4.31)$$

where the function $H(z, A)$ is given by

$$H(z, A) = \sqrt{\left(1 - \frac{z}{L}\right)^2 + A^2} - \sqrt{\left(\frac{z}{L}\right)^2 + A^2} - \left|1 - \frac{z}{L}\right| + \left|\frac{z}{L}\right|, \quad A = R/L. \quad (4.32)$$

Using Maxwell's equations allows one to find E_r . For cylindrical symmetry, Gauss's law takes the form

$$\frac{\partial}{\partial r}(rE_r) = r \left(\frac{\rho_0}{\epsilon_0} - \frac{\partial E_z}{\partial z} \right),$$

where the charge density $\rho = \rho_0[\theta_H(r) - \theta_H(r - R)] \cdot [\theta_H(z) - \theta_H(z - L)]$. Approximating Gauss's law to first order in r gives

$$\frac{\partial}{\partial r}(rE_r) = r \left(\frac{\rho_0}{\epsilon_0} [\theta_H(z) - \theta_H(z - L)] - \frac{dE_z}{dz}(r=0) \right). \quad (4.33)$$

The derivative dE_z/dz is given by

$$\frac{d}{dz}E_z(0, z) = \frac{\rho_0}{2\epsilon_0} \left(\frac{z - L}{\sqrt{(z - L)^2 + R^2}} - \frac{z}{\sqrt{z^2 + R^2}} + 2[\theta_H(z) - \theta_H(z - L)] \right). \quad (4.34)$$

Plugging this expression into Gauss's law then yields

$$\frac{\partial}{\partial r}(rE_r) \cong r \frac{\rho_0}{2\epsilon_0} \left(\frac{L - z}{\sqrt{(L - z)^2 + R^2}} + \frac{z}{\sqrt{z^2 + R^2}} \right). \quad (4.35)$$

Thus the radial component is given to first order in r as

$$E_r = r \frac{\rho_0}{4\epsilon_0} \left(\frac{L-z}{\sqrt{(L-z)^2 + R^2}} + \frac{z}{\sqrt{z^2 + R^2}} \right).$$

This can be written as

$$E_r = \frac{Qr}{4\pi\epsilon_0 R^2 L} \left(\frac{1-z/L}{\sqrt{(1-z/L)^2 + (R/L)^2}} + \frac{z/L}{\sqrt{(z/L)^2 + (R/L)^2}} \right) \quad (4.36)$$

This is written more compactly as

$$E_r = \frac{Qr}{4\pi\epsilon_0 R^2 L} G(z, A). \quad (4.37)$$

where $G(z, A)$ is the term in parenthesis and $A = R/L$ is the aspect ratio of the cylinder. For a moving cylinder of charge, these fields must be boosted back to the lab frame. This is accomplished using Eq. (4.10). If the cylinder is moving down the z -axis with a velocity given by $\dot{z} = c\beta$, the transformation is the same as before, except now there is a z component to the electric field $E_z = E'_z$. Remembering to include the contraction effect on the length of the cylinder ($z \rightarrow \gamma z$ and $L \rightarrow \gamma L$), the fields can be written as

$$E_z = \frac{Q}{2\pi\epsilon_0 R^2} H(\zeta, A), \quad E_r = \frac{Qr}{4\pi\epsilon_0 R^2 L} G(\zeta, A), \quad B_\theta = \frac{\beta}{c} E_r, \quad (4.38)$$

where the tail of the cylinder is z_t and $\zeta = z - z_t$. The functions H and G have been previously defined. In this expression $A = R/\gamma L$.

The envelope equation can be written down for the propagation for each slice by assuming the space charge force from the bunch is approximately the same as the space charge force from a uniform cylinder with charge and length given by the full bunch charge and length, but with a radius given by the slice radius. Identifying the $\lambda = Q/L'$ in the beam rest frame, the first order component of the transverse fields can be written as

$$E_r^{(1)} = \frac{\lambda' r}{8\pi\epsilon_0 \sigma^2} \left[\frac{G(\zeta', A')}{2} \right]. \quad (4.39)$$

Following the previous analysis, the envelope equation then becomes

$$\ddot{\sigma} + [\gamma^2 \beta \dot{\beta}] \dot{\sigma} + \left[\frac{K}{m\gamma} + (\dot{\theta}_r)^2 \right] \sigma = \left[\frac{c^2 k_p}{\beta \gamma^3} \left(\frac{G(\zeta, A)}{2} \right) \right] \frac{1}{\sigma} + \left(\frac{c\epsilon_n}{\gamma} \right)^2 \frac{1}{\sigma^3},$$

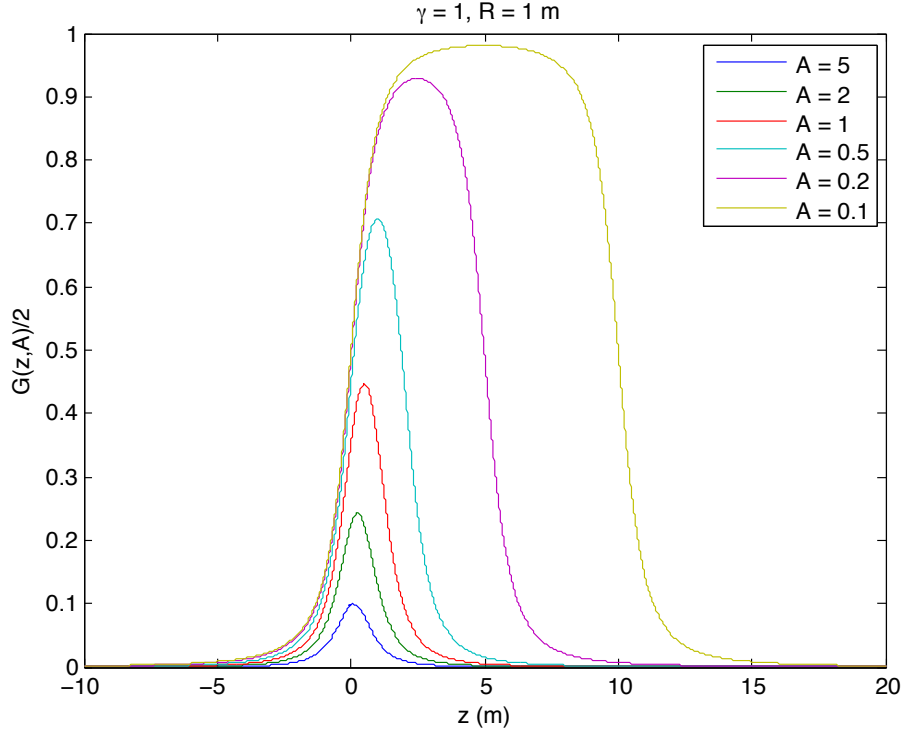
Explicitly, the perverance term is written out as

$$K = \frac{I}{2I_0} \cdot \frac{G}{2} = c\beta(\zeta) \left(\frac{Q}{L} \right) \left(\frac{G(\zeta, A)}{2} \right). \quad (4.40)$$

This is exactly the form for the current described in the previous section. The function $G(\zeta, A)/2$ is identified as the form factor term $g(\zeta)$ in Eq. (4.28). Fig. 4.2 shows the the function $G(\zeta, A)/2$ for various values of the aspect ratio $A = R/L/\gamma$. As the figure shows the form factor is less than or equal to unity.

It turns out that this form factor can also be used to describe elliptical beams as well. For long beams, the space-charge fields on envelope from an elliptical beam are equivalent to those of a circular beam with a effective radius $R^* = (X + Y)/2$. This equivalence gives a good approximation for bunched beams as well [88]. For writing down the final envelope equations, the effect of image charges (for emission from a cathode) are included. This is done by letting

$$\frac{I(\zeta)g(\zeta)}{\gamma^2} \rightarrow I(\zeta)[(1 - \beta^2)g(\zeta) - (1 + \beta^2)g(\xi)],$$

Figure 4.2: The bunch form factor $g(\zeta) = G(\zeta, A)/2$.

where $\xi = z_s + z_h$. Effectively, this just includes a mirror bunch behind the cathode. With this, the final equations of motion for the HOMDYN algorithm are written collected together. For the case of a cylindrically system, the equations are given in the Larmor frame as:

$$\ddot{\sigma}_L + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_L + \left[\frac{K}{m\gamma} + (\dot{\theta}_r)^2 \right] \sigma_L = \frac{c^2 k_p}{2\beta\gamma\sigma_L} [\gamma^{-2} G(\zeta, A) - (1 + \beta^2) G(\xi, A)] + \left(\frac{c\varepsilon_n}{\gamma} \right)^2 \frac{1}{\sigma_L^3}. \quad (4.41)$$

For an elliptical beam in an uncoupled focusing channel the equations are

$$\ddot{\sigma}_i + [\gamma^2 \beta \dot{\beta}] \dot{\sigma}_i + \left[\frac{K_i}{m\gamma} \right] \sigma_i = \frac{c^2 k_p}{2\beta\gamma\sigma^*} [\gamma^{-2} G(\zeta, A^*) - (1 + \beta^2) G(\xi, A^*)] + \left(\frac{c\varepsilon_{n,i}}{\gamma} \right)^2 \frac{1}{\sigma_i^3}. \quad (4.42)$$

In this expression $\sigma^* = (\sigma_x + \sigma_y)/2$ and $A^* = R^*/\gamma L$. These equations can be cast in their equivalent form in terms of the beam sizes. This is done below:

$$\ddot{R} + [\gamma^2 \beta \dot{\beta}] \dot{R} + \left[\frac{K}{m\gamma} + (\dot{\theta}_r)^2 \right] R = \frac{2c^2 k_p}{\beta\gamma R} [\gamma^{-2} G(\zeta, A) - (1 + \beta^2) G(\xi, A)] + \left(\frac{4c\varepsilon_{n,x}}{\gamma} \right)^2 \frac{1}{R^3}, \quad (4.43)$$

For an elliptical beam in an uncoupled focusing channeling the equations are

$$\ddot{X}_i + [\gamma^2 \beta \dot{\beta}] \dot{X}_i + \left[\frac{K_i}{m\gamma} \right] X_i = \frac{2c^2 k_p}{\beta \gamma R^*} [\gamma^{-2} G(\zeta, A^*) - (1 + \beta^2) G(\xi, A^*)] + \left(\frac{4c\varepsilon_{n,i}}{\gamma} \right)^2 \frac{1}{X_i^3}. \quad (4.44)$$

In all of these expressions the kinetic functions β and γ as well as the beam sizes are functions of the slice position ζ .

The BET Model: Analytic Space Charge Fields for Bunched Beams

4.4.3 Code Comparison

So far the envelope tracker using the HOMDYN space charge model has been successfully compared with the OPAL-t code for the first 12 meters of the 250 MeV test injector at PSI. The results for the both the transverse and longitudinal rms beam sizes and the emittance are shown in Fig. 4.3 As both plots show, the agreement between

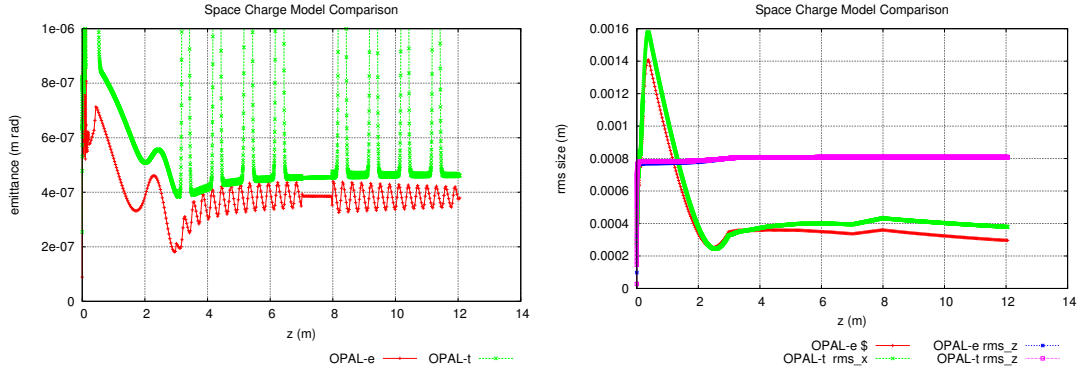


Figure 4.3: Comparison of the OPAL-e envelope tracker and the full 3D OPAL-t tracker in the first 12 meters for the 250 MeV test injector at PSI. Shown here are both σ_x and σ_z (a), and the emittance (b).

both trackers is quite good. To generate this data, the envelope tracker was run with 100 beam slices.

4.5 Space Charge

Space-charge effects will be included in the simulation by specifying a field solver described in Chapter 12 and using the solver in the track command described in Chapter 15. By default, the code does not assume any symmetry i.e. full 3D. In the near future it is planned to implement also a slice (2D) model. This will allow the use of less numbers of macro-particles in the simulation which reduces the computational time significantly.

The space-charge forces are calculated by solving the 3D Poisson equation with open boundary conditions using a standard or integrated Green function method. The image charge effects of the conducting cathode are also included using a shifted Green function method. If more than one Lorentz frame is defined, the total space-charge forces are then the summation of contributions from all Lorentz frames. More details will be given in Version 1.6.0

4.5.1 FFT Based Particle-Mesh (PM) Solver

The Particle-Mesh (PM) solver is one of the oldest improvements over the PP solver. Still one of the best references is the book by R.W. Hockney & J.W. Eastwood [73]. The PM solver introduces a discretisation of space. The

rectangular computation domain $\Omega := [-L_x, L_x] \times [-L_y, L_y] \times [-L_t, L_t]$, just big enough to include all particles, is segmented into a regular mesh of $M = M_x \times M_y \times M_t$ grid points. For the discussion below we assume $N = M_x = M_y = M_t$.

The solution of Poisson's equation is an essential component of any self-consistent electrostatic beam dynamics code that models the transport of intense charged particle beams in accelerators. If the bunch is small compared to the transverse size of the beam pipe, the conducting walls are usually neglected. In such cases the Hockney method may be employed [73, 74, 75]. In that method, rather than computing N_p^2 point-to-point interactions (where N_p is the number of macroparticles), the potential is instead calculated on a grid of size $(2N)^d$, where N is the number of grid points in each dimension of the physical mesh containing the charge, and where d is the dimension of the problem. Using the Hockney method, the calculation is performed using Fast Fourier Transform (FFT) techniques, with the computational effort scaling as $(2N)^d (\log_2 2N)^d$.

When the beam bunch fills a substantial portion of the beam pipe transversely, or when the bunch length is long compared with the pipe transverse size, the conducting boundaries cannot be ignored. Poisson solvers have been developed previously to treat a bunch of charge in an open-ended pipe with various geometries [76, 77].

The solution of the Poisson equation,

$$\nabla^2 \phi = -\rho/\epsilon_0, \quad (4.45)$$

for the scalar potential, ϕ , due to a charge density, ρ , and appropriate boundary conditions, can be expressed as,

$$\phi(x, y, z) = \int \int \int dx' dy' dz' \rho(x', y', z') G(x, x', y, y', z, z'), \quad (4.46)$$

where $G(x, x', y, y', z, z')$ is the Green function, subject to the appropriate boundary conditions, describing the contribution of a source charge at location (x', y', z') to the potential at an observation location (x, y, z) .

For an isolated distribution of charge this reduces to

$$\phi(x, y, z) = \int \int \int dx' dy' dz' \rho(x', y', z') G(x - x', y - y', z - z'), \quad (4.47)$$

where

$$G(u, v, w) = \frac{1}{\sqrt{u^2 + v^2 + w^2}}. \quad (4.48)$$

A simple discretisation of Eq. (4.47) on a Cartesian grid with cell size (h_x, h_y, h_z) leads to,

$$\phi_{i,j,k} = h_x h_y h_z \sum_{i'=1}^{M_x} \sum_{j'=1}^{M_y} \sum_{k'=1}^{M_t} \rho_{i',j',k'} G_{i-i',j-j',k-k'}, \quad (4.49)$$

where $\rho_{i,j,k}$ and $G_{i-i',j-j',k-k'}$ denote the values of the charge density and the Green function, respectively, defined on the grid M .

FFT-based Convolutions and Zero Padding

FFT's can be used to compute convolutions by appropriate zero-padding of the sequences. Discrete convolutions arise in solving the Poisson equation, and one is typically interested in the following,

$$\begin{aligned} j &= 0, \dots, J-1 \\ \bar{\phi}_j &= \sum_{k=0}^{K-1} \bar{\rho}_k \bar{G}_{j-k} \quad , \quad k = 0, \dots, K-1 \\ j-k &= -(K-1), \dots, J-1 \end{aligned} \quad (4.50)$$

where \bar{G} corresponds to the free space Green function, $\bar{\rho}$ corresponds to the charge density, and $\bar{\phi}$ corresponds to the scalar potential. The sequence $\{\bar{\phi}_j\}$ has J elements, $\{\bar{\rho}_k\}$ has K elements, and $\{\bar{G}_m\}$ has $M = J + K - 1$ elements.

One can zero-pad the sequences to a length $N \geq M$ and use FFTs to efficiently obtain the $\{\bar{\phi}_j\}$ in the unpadded region. This defines a zero-padded charge density, ρ ,

$$\rho_k = \begin{cases} \bar{\rho}_k & \text{if } k = 0, \dots, K-1 \\ 0 & \text{if } k = K, \dots, N-1. \end{cases} \quad (4.51)$$

Define a periodic Green function, G_m , as follows,

$$G_m = \begin{cases} \bar{G}_m & \text{if } m = -(K-1), \dots, J-1 \\ 0 & \text{if } m = J, \dots, N-K, \\ G_{m+iN} = G_m & \text{for } i \text{ integer.} \end{cases} \quad G \quad (4.52)$$

Now consider the sum

$$\phi_j = \frac{1}{N} \sum_{k=0}^{N-1} W^{-jk} \left(\sum_{n=0}^{N-1} \rho_n W^{nk} \right) \left(\sum_{m=0}^{N-1} G_m W^{mk} \right), \quad 0 \leq j \leq N-1, \quad (4.53)$$

where $W = e^{-2\pi i/N}$. This is just the FFT-based convolution of $\{\rho_k\}$ with $\{G_m\}$. Then,

$$\phi_j = \sum_{n=0}^{K-1} \sum_{m=0}^{N-1} \bar{\rho}_n G_m \frac{1}{N} \sum_{k=0}^{N-1} W^{(m+n-j)k} \quad 0 \leq j \leq N-1. \quad (4.54)$$

Now use the relation

$$\sum_{k=0}^{N-1} W^{(m+n-j)k} = N \delta_{m+n-j, iN} \quad (i \text{ an integer}). \quad (4.55)$$

It follows that

$$\phi_j = \sum_{n=0}^{K-1} \bar{\rho}_n G_{j-n+iN} \quad 0 \leq j \leq N-1. \quad (4.56)$$

But G is periodic with period N . Hence,

$$\phi_j = \sum_{n=0}^{K-1} \bar{\rho}_n G_{j-n} \quad 0 \leq j \leq N-1. \quad (4.57)$$

In the physical (unpadded) region, $j \in [0, J-1]$, so the quantity $j-n$ in Eq. (4.57) satisfies $-(K-1) \leq j-n \leq J-1$. In other words the values of G_{j-n} are identical to \bar{G}_{j-n} . Hence, in the physical region the FFT-based convolution, Eq. (4.53), matches the convolution in Eq. (4.50).

As stated above, the zero-padded sequences need to have a length $N \geq M$, where M is the number of elements in the Green function sequence $\{x_m\}$. In particular, one can choose $N = M$, in which case the Green function sequence is not padded at all, and only the charge density sequence, $\{r_k\}$, is zero-padded, with $k = 0, \dots, K-1$ corresponding to the physical region and $k = K, \dots, M-1$ corresponding to the zero-padded region.

The above FFT-based approach – zero-padding the charge density array, and circular-shifting the Green function in accordance with Eq. (4.52) – will work in general. In addition, if the Green function is a symmetric function of its arguments, the value at the end of the Green function array (at grid point $J-1$) can be dropped, since it will be

recovered implicitly through the symmetry of Eq. (4.52). In that case the approach is identical to the Hockney method [73, 74, 75].

Lastly, note that the above proof that the convolution, Eq. (4.53), is identical to Eq. (4.50) in the unpadded region, works even when W^{-jk} and W^{mk} are replaced by W^{jk} and W^{-mk} , respectively, in Eq. (4.53). In other words, the FFT-based approach can be used to compute

$$\begin{aligned} j &= 0, \dots, J-1 \\ \bar{\phi}_j &= \sum_{k=0}^{K-1} \bar{\rho}_k \bar{G}_{j+k} \quad , \quad k = 0, \dots, K-1 \\ j-k &= -(K-1), \dots, J-1 \end{aligned} \tag{4.58}$$

simply by changing the direction of the Fourier transform of the Green function and changing the direction of the final Fourier transform.

Algorithm used in OPAL

As a result, the solution of Eq. (4.49) is then given by

$$\phi_{i,j,k} = h_x h_y h_z \text{FFT}^{-1}\{(\text{FFT}\{\rho_{i,j,k}\})(\text{FFT}\{G_{i,j,k}\})\} \tag{4.59}$$

where the notation has been introduced that $\text{FFT}\{.\}$ denotes a forward FFT in all 3 dimensions, and $\text{FFT}^{-1}\{.\}$ denotes a backward FFT in all 3 dimensions.

Interpolation Schemes

More details will be given in Version 1.6.0

Iterative Space Charge Solver

This is a scalable parallel solver for the Poisson equation within a Particle-In-Cell (PIC) code for the simulation of electron beams in particle accelerators of irregular shape. The problem is discretized by Finite Differences. Depending on the treatment of the Dirichlet boundary the resulting system of equations is symmetric or ‘mildly’ nonsymmetric positive definite. In all cases, the system is solved by the preconditioned conjugate gradient algorithm with smoothed aggregation (SA) based algebraic multigrid (AMG) preconditioning. More details are given in [52].

Energy Binning

The beam out of a cathode or in a plasma wake field accelerator can have a large energy spread. In this case, the static approximation using one Lorentz frame might not be sufficient. Multiple Lorentz frames can be used so that within each Lorentz frame the energy spread is small and hence the electrostatic approximation is valid. More details will be given in Version 1.6.0

4.6 Wake Fields

Longitudinal and transverse short range wakefields wake fields are described in Chapter 13.

4.7 Multiple Species

In the present version only one particle species can be defined (Chapter 10), however due to the underlying general structure, the implementation of a true multi species version of OPAL is trivial.

Chapter 5

OPAL-CYCL

5.1 Introduction

OPAL-CYCL, as one of the flavors of the OPAL framework, is a fully three-dimensional parallel beam dynamics simulation program dedicated to future high intensity cyclotrons and FFAG. It tracks multiple particles which takes into account the space charge effects. For the first time in the cyclotron community, OPAL-CYCL has the capability of tracking multiple bunches simultaneously and take into account the beam-beam effects of the radially neighboring bunches (we call it neighboring bunch effects for short) by using a self-consistent numerical simulation model.

Apart from the multiparticle simulation mode, OPAL-CYCL also has two other serial tracking modes for conventional cyclotron machine design. One mode is the single particle tracking mode, which is a useful tool for the preliminary design of a new cyclotron. It allows one to compute basic parameters, such as reference orbit, phase shift history, stable region, and matching phase ellipse. The other one is the tune calculation mode, which can be used to compute the betatron oscillation frequency. This is useful for evaluating the focusing characteristics of a given magnetic field map.

In additions, the widely used plugin elements, including collimator, radial profile probe, septum, trim-coil field and charge stripper, are currently implemented in OPAL-CYCL. These functionalities are very useful for designing, commissioning and upgrading of cyclotrons and FFAGs.

5.2 Tracking modes

According to the number of particles defined by the argument `NPART` in `BEAM` (see Section 10.1), OPAL-CYCL works in one of the following three operation modes automatically.

5.2.1 Single Particle Tracking mode

In this mode, only one particle is tracked, either with acceleration or not. Working in this mode, OPAL-CYCL can be used as a tool during the preliminary design phase of a cyclotron.

The 6D parameters of a single particle in the initial local frame must be read from a file. To do this, in the OPAL input file, the command line `DISTRIBUTION` (see Section 11) should be defined like this:

```
Dist1: DISTRIBUTION, DISTRIBUTION=fromfile, FNAME="PartDatabase.dat";
```

where the file *PartDatabase.dat* should have two lines:

```
1
0.001 0.001 0.001 0.001 0.001 0.001
```

The number in the first line is the total number of particles. In the second line the data represents x, p_x, y, p_y, z, p_z in the local reference frame. Their units are described in Section 5.3.

Please don't try to run his mode in parallel environment. You should believe that a single processor of the 21st century is capable of doing the single particle tracking.

5.2.2 Tune Calculation mode

In this mode, two particles are tracked, one with all data is set to zero is the reference particle and another one is an off-centering particle which is off-centered in both r and z directions. Working in this mode, OPAL-CYCL can calculate the betatron oscillation frequency ν_r and ν_z for different energies to evaluate the focusing characteristics for a given magnetic field.

Like the single particle tracking mode, the initial 6D parameters of the two particles in the local reference frame must be read from a file, too. In the file should has three lines:

2					
0.0	0.0	0.0	0.0	0.0	0.0
0.001	0.0	0.0	0.0	0.001	0.0

When the total particle number equals 2, this mode is triggered automatically. Only the element CYCLOTRON in the beam line is used and other elements are omitted if any exists.

Please don't try to run his mode in parallel environment, either.

5.2.3 Multi-particle tracking mode

In this mode, large scale particles can be tracked simultaneously, either with space charge or not, either single bunch or multi-bunches, either serial or parallel environment, either reading the initial distribution from a file or generating a typical distribution, either running from the beginning or restarting from the last step of a former simulation.

Because this is the main mode as well as the key part of OPAL-CYCL, we will describe this in detail in Section 5.8.

5.3 Variables in OPAL-CYCL

OPAL-CYCL uses the following canonical variables to describe the motion of particles:

X Horizontal position x of a particle in given global Cartesian coordinates [m].

PX Horizontal canonical momentum [eV/c].

Y Longitudinal position y of a particle in global Cartesian coordinates [m].

PY Longitudinal canonical momentum [eV/c].

Z Vertical position z of a particle in global Cartesian coordinates [m].

PZ Vertical canonical momentum [eV/c].

The independent variable is: **t** [s].

NOTE: unit conversion of momentum in OPAL-T and OPAL-CYCL

Convert $\beta_x \gamma$ [dimensionless] to [mrad],

$$(\beta \gamma)_{\text{ref}} = \frac{P}{m_0 c} = \frac{P c}{m_0 c^2}, \quad (5.1)$$

$$P_x [\text{mrad}] = 1000 \times \frac{(\beta_x \gamma)}{(\beta \gamma)_{\text{ref}}}. \quad (5.2)$$

Convert from [eV/c] to $\beta_x \gamma$ [dimensionless],

$$(\beta_x \gamma) = \sqrt{\left(\frac{P_x [\text{eV}/c]}{m_0 c} + 1\right)^2 - 1}. \quad (5.3)$$

This may be deduced by analogy for the y and z directions.

5.3.1 The initial distribution in the local reference frame

To ensure compatibility with OPAL-T the initial distribution of the bunch, either read from file or generated by a distribution generator (see Section 11), is specified in the local reference frame which is in Cartesian coordinates. At the beginning of the run, the 6 phase space variables (x, y, z, p_x, p_y, p_z) are transformed to the global Cartesian coordinates using the starting coordinates r_0 (RINIT), ϕ_0 (PHIINIT), and z_0 (ZINIT), and the starting momenta p_{r0} (PRINIT), and p_{z0} (PZINIT) of the reference particle, defined in the CYCLOTRON element (see Section 8.10). Note that $p_{\phi 0}$ is calculated automatically from p_{total} , p_{r0} , and p_{z0} .

$$X = x \cos(\phi_0) - y \sin(\phi_0) + r_0 \cos(\phi_0)$$

$$Y = x \sin(\phi_0) + y \cos(\phi_0) + r_0 \sin(\phi_0)$$

$$Z = z + z_0$$

$$PX = (p_x + p_{r0}) \cos(\phi_0) - (p_y + p_{\phi 0}) \sin(\phi_0)$$

$$PY = (p_x + p_{r0}) \sin(\phi_0) + (p_y + p_{\phi 0}) \cos(\phi_0)$$

$$PZ = p_z + p_{z0}$$

5.4 Field Maps

In OPAL-CYCL, the magnetic field on the median plane is read from an ASCII type file. The field data should be stored in the cylinder coordinates frame (because the field map on the median plane of the cyclotron is usually measured in this frame).

There are two possible situations. One is the real field map on median plane of the exist cyclotron machine using measurement equipment. Limited by the narrow gaps of magnets, in most cases with cyclotrons, only vertical field B_z on the median plane ($z = 0$) is measured. Since the magnetic field data off the median plane field components is necessary for those particles with $z \neq 0$, the field need to be expanded in Z direction. According to the approach given by Gordon and Taivassalo, by using a magnetic potential and measured B_z on the median plane, at the point (r, θ, z) in cylindrical polar coordinates, the 3th order field can be written as

$$\begin{aligned} B_r(r, \theta, z) &= z \frac{\partial B_z}{\partial r} - \frac{1}{6} z^3 C_r, \\ B_\theta(r, \theta, z) &= \frac{z}{r} \frac{\partial B_z}{\partial \theta} - \frac{1}{6} \frac{z^3}{r} C_\theta, \\ B_z(r, \theta, z) &= B_z - \frac{1}{2} z^2 C_z, \end{aligned} \quad (5.4)$$

where $B_z \equiv B_z(r, \theta, 0)$ and

$$\begin{aligned} C_r &= \frac{\partial^3 B_z}{\partial r^3} + \frac{1}{r} \frac{\partial^2 B_z}{\partial r^2} - \frac{1}{r^2} \frac{\partial B_z}{\partial r} + \frac{1}{r^2} \frac{\partial^3 B_z}{\partial r \partial \theta^2} - 2 \frac{1}{r^3} \frac{\partial^2 B_z}{\partial \theta^2}, \\ C_\theta &= \frac{1}{r} \frac{\partial^2 B_z}{\partial r \partial \theta} + \frac{\partial^3 B_z}{\partial r^2 \partial \theta} + \frac{1}{r^2} \frac{\partial^3 B_z}{\partial \theta^3}, \\ C_z &= \frac{1}{r} \frac{\partial B_z}{\partial r} + \frac{\partial^2 B_z}{\partial r^2} + \frac{1}{r^2} \frac{\partial^2 B_z}{\partial \theta^2}. \end{aligned} \quad (5.5)$$

All the partial differential coefficients are on the median plane and can be calculated by interpolation. OPAL-CYCL uses Lagrange's 5-point formula.

The other situation is to calculate the field on the median plane or the 3D fields of the working gap for interesting region numerically by creating 3D model using commercial software, such as TOSCA, ANSOFT and ANSYS during the design phase of a new machine. If the field on the median plane is calculated, the field off the median plane can be obtained using the same expansion approach as the measured field map as described above. However, the 3D fields of the entire working gap should be more accurate than the expansion approach especially at the region not so close to the median plane in Z direction.

In the current version, we implemented the three specific type field-read functions *Cyclotron :: getFieldFromFile()* of the median plane fields. That which function is used is controlled by the parameters TYPE of CYCLOTRON (see Section 8.10) in the input file.

5.4.1 CARBONCYCL type

If TYPE=CARBONCYCL, the program requires the B_z data which is stored in a sequence shown in Fig.5.1. We

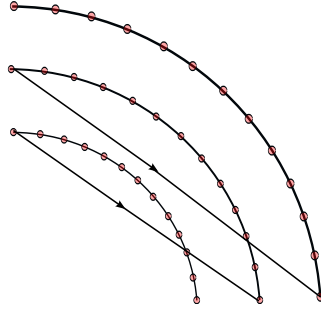


Figure 5.1: 2D field map on the median plane with primary direction corresponding to the azimuthal direction, secondary direction to the radial direction

need to add 6 parameters at the header of a plain B_z (kGauss) data file, namely, r_{min} [mm], Δr [mm], θ_{min} [°], $\Delta\theta$ [°], N_θ (total data number in each arc path of azimuthal direction) and N_r (total path number along radial direction). If Δr or $\Delta\theta$ is decimal, one can set its negative opposite number. For instance, if $\Delta\theta = \frac{1}{3}^\circ$, the fourth line of the header should be set to -3.0. Example showing the above explained format:

```

3.0e+03
10.0
0.0
-3.0
300
161
1.414e-03  3.743e-03  8.517e-03  1.221e-02  2.296e-02
3.884e-02  5.999e-02  8.580e-02  1.150e-01  1.461e-01
1.779e-01  2.090e-01  2.392e-01  2.682e-01  2.964e-01
```

```
3.245e-01  3.534e-01  3.843e-01  4.184e-01  4.573e-01
.....
```

5.4.2 CYCIAE type

If TYPE=CYCIA, the program requires data format given by ANSYS 10.0. This function is originally for the 100 MeV cyclotron of CIAE, whose isochronous fields is numerically computed by by ANSYS. The median plane fields data is output by reading the APDL (ANSYS Parametric Design Language) script during the post-processing phase (you may need to do minor changes to adapt your own cyclotron model):

```
/post1
resume,solu,db
csys,1
nsel,s,loc,x,0
nsel,r,loc,y,0
nsel,r,loc,z,0
PRNSOL,B,COMP

CSYS,1
rsys,1

*do,count,0,200
  path,cyc100_Ansys,2,5,45
  ppath,1,,0.01*count,0,,1
  ppath,2,,0.01*count/sqrt(2),0.01*count/sqrt(2),,1

  pdef,bz,b,z
  paget,data,table

  *if,count,eq,0,then
    /output,cyc100_ANSYS,dat
    *STATUS,data,,,5,5
    /output
  *else
    /output,cyc100_ANSYS,dat,,append
    *STATUS,data,,,5,5
    /output
  *endif
*enddo
finish
```

By running this in ANSYS, you can get a fields file with the name *cyc100_ANSYS.data*. You need to put 6 parameters at the header of the file, namely, r_{min} [mm], Δr [mm], θ_{min} [°], $\Delta\theta$ [°], N_θ (total data number in each arc path of azimuthal direction) and N_r (total path number along radial direction). If Δr or $\Delta\theta$ is decimal, one can set its negative opposite number. This is useful if the decimal is unlimited. For instance, if $\Delta\theta = \frac{1}{3}^\circ$, the fourth line of the header should be -3.0. In a word, the fields are stored in the following format:

```
0.0
10.0
0.0e+00
1.0e+00
90
201
PARAMETER STATUS- DATA  ( 336 PARAMETERS DEFINED)
                        (INCLUDING    17 INTERNAL PARAMETERS)

LOCATION              VALUE
   1         5         1  0.537657876
   2         5         1  0.538079473
```

```

      3      5      1  0.539086731
      .....
     44      5      1  0.760777286
     45      5      1  0.760918663
     46      5      1  0.760969074

PARAMETER STATUS- DATA  ( 336 PARAMETERS DEFINED)
                        (INCLUDING 17 INTERNAL PARAMETERS)

LOCATION              VALUE
  1      5      1  0.704927299
  2      5      1  0.705050993
  3      5      1  0.705341275
      .....

```

5.4.3 BANDRF type

If TYPE=BANDRF, the program requires the B_z data format which is same as CARBONCYCL. But with BANDRF type, the program can also read in the 3D electric field(s). For the detail about its usage, please see Section 8.10.

5.4.4 Default PSI format

If the value of TYPE is other string rather than above mentioned, the program requires the data format like PSI format field file ZYKL9Z.NAR and SO3AV.NAR, which was given by the measurement. We add 4 parameters at the header of the file, namely, r_{min} [mm], Δr [mm], θ_{min} [°], $\Delta\theta$ [°], If Δr or $\Delta\theta$ is decimal, one can set its negative opposite number. This is useful if the decimal is unlimited. For instance, if $\Delta\theta = \frac{1}{3}^\circ$, the fourth line of the header should be -3.0.

```

1900.0
20.0
0.0
-3.0
LABEL=SO3AV
CFELD=FIELD      NREC= 141      NPAR= 3
LPAR= 7          IENT= 1        IPAR= 1
                3          141      135      30      8
                8          70
LPAR= 1089      IENT= 2        IPAR= 2
0.100000000E+01 0.190000000E+04 0.200000000E+02 0.000000000E+00 0.333333343E+00
0.506500015E+02 0.600000000E+01 0.938255981E+03 0.100000000E+01 0.240956593E+01
0.282477260E+01 0.340503168E+01 0.419502926E+01 0.505867147E+01 0.550443363E+01
0.570645094E+01 0.579413509E+01 0.583940887E+01 0.586580372E+01 0.588523722E+01
      .....

```

5.4.5 3D fieldmap

It is additionally possible to load 3D fieldmaps for tracking through OPAL-CYCL. 3D fieldmaps are loaded by sequentially adding new field elements to a line, as is done in OPAL-T. It is not possible to add RF cavities while operating in this mode. In order to define ring parameters such as initial ring radius a RINGDEFINITION type is loaded into the line, followed by one or more SBEND3D elements.

```

ringdef: RINGDEFINITION, HARMONIC_NUMBER=6, LAT_RINIT=2350.0, LAT_PHIINIT=0.0,
        LAT_THETAINIT=0.0, BEAM_PHIINIT=0.0, BEAM_PRINIT=0.0,
        BEAM_RINIT=2266.0, SYMMETRY=4.0, RFFREQ=0.2;

triplet: SBEND3D, FMAPFN="fdf-tosca-field-map.table", LENGTH_UNITS=10., FIELD_UNITS=-1e-4;

```

```
11: Line = (ringdef, triplet, triplet);
```

The fieldmap with filename `fdf-tosca-field-map.table` is loaded, which is a file like

```

      422280      422280      422280      1
1 X [LENGU]
2 Y [LENGU]
3 Z [LENGU]
4 BX [FLUXU]
5 BY [FLUXU]
6 BZ [FLUXU]
0
194.01470 0.0000000 80.363520 0.68275932346E-07 -5.3752492577 0.28280706805E-07
194.36351 0.0000000 79.516210 0.42525693524E-07 -5.3827955117 0.17681348191E-07
194.70861 0.0000000 78.667380 0.19766168358E-07 -5.4350026348 0.82540823165E-08
<continues>
```

The header parameters are ignored - user supplied parameters `LENGTH_UNITS` and `FIELD_UNITS` are used. Fields are supplied on points in a grid in (r, y, ϕ) . Positions and field elements are specified by Cartesian coordinates (x, y, z) .

5.4.6 user's own fieldmap

You should revise the function or write your own function according to the instructions in the code to match your own field format if it is different to above types. For more detail about the parameters of `CYCLOTRON`, please refer to Section 8.10.

5.5 RF field

5.5.1 Read RF voltage profile

The RF cavities are treated as straight lines with infinitely narrow gaps and the electric field is a δ function plus a transit time correction. the two-gap cavity is treated as two independent single-gap cavities. the spiral gap cavity is not implemented yet. For more detail about the parameters of cyclotron cavities, see Section 8.12.2.

The voltage profile of a cavity gap is read from ASCII file. Here is an example:

```

6
0.00      0.15      2.40
0.20      0.65      2.41
0.40      0.98      0.66
0.60      0.88     -1.59
0.80      0.43     -2.65
1.00     -0.05     -1.71
```

The number in the first line means 6 sample points and in the following lines the three values represent the normalized distance to the inner edge of the cavity, the normalized voltage and its derivative respectively.

5.5.2 Read 3D RF fieldmap

The 3D RF fieldmap can be read from H5hut type file. This is useful for modeling the central region electric fields which usually has complicate shapes. For the detail about its usage, please see Section 8.10.

Please note that in this case, the E field is treated as a part of `CYCLOTRON` element, rather than a independent `RFCAVITY` element.

5.6 Particle Tracking and Acceleration

The precision of the tracking methods is vital for the entire simulation process, especially for long distance tracking jobs. OPAL-CYCL uses a 4th order Runge-Kutta algorithm and the second order Leap-Frog scheme. The 4th order Runge-Kutta algorithm needs 4 the external magnetic field evaluation in each time step τ . During the field interpolation process, for an arbitrary given point the code first interpolates Formula B_z for its counterpart on the median plane and then expands to this give point using (5.4).

After each time step i , the code detects whether the particle crosses any one of the RF cavities during this step. If it does, the time point t_c of crossing is calculated and the particle return back to the start point of step i . Then this step is divided into three substeps: first, the code tracks this particle for $t_1 = \tau - (t_c - t_{i-1})$; then it calculates the voltage and adds momentum kick to the particle and refreshes its relativistic parameters β and γ ; and then tracks it for $t_2 = \tau - t_1$.

5.7 Space Charge

OPAL-CYCL uses the same solvers as OPAL-T to calculate the space charge effects (see Section 4.5). The difference is that, in a cyclotron, both the origin and orientation of the local Cartesian frame are changed from time to time. So the coordinates are transformed from the global frame to the local frame first, then the space charge fields are calculated in the local frame. After the space charge fields are solved, both coordinates and self electric and magnetic field are transformed back to global frame.

Typically, the space charge field is calculated once per time step. This is no surprise for the second-order Boris-Buneman time integrator (leapfrog-like scheme) which has per default only one force evaluation per step. The fourth-order Runge-Kutta integrator keeps the space charge field constant for one step, although there are four external field evaluations. There is an experimental multiple-time-stepping (MTS) variant of the Boris-Buneman/leapfrog-method, which evaluates space charge only every N -th step, thus greatly reducing computation time while usually being still accurate enough.

5.8 Multi-bunches Issues

The neighboring bunches problem is motivated by the fact that for high intensity cyclotrons with small turn separation, single bunch space charge effects are not the only contribution. Along with the increment of beam current, the mutual interaction of neighboring bunches in radial direction becomes more and more important, especially at large radius where the distances between neighboring bunches get increasingly small and even they can overlap each other. One good example is PSI 590 MeV Ring cyclotron with a current of about 2mA in CW operation and the beam power amounts to 1.2 MW. A upgrade project for Ring is in process with the goal of 1.8 MW CW on target by replacing four old aluminum resonators by four new copper cavities with peak voltage increasing from about 0.7 MV to above 0.9 MV. After upgrade, the total turn number is reduced from 200 turns to less then 170 turns. Turn separation is increased a little bit, but still are at the same order of magnitude as the radial size of the bunches. Hence once the beam current increases from 2 mA to 3 mA, the mutual space charge effects between radially neighboring bunches can have significant impact on beam dynamics. In consequence, it is important to cover neighboring bunch effects in the simulation to quantitatively study its impact on the beam dynamics.

In OPAL-CYCL we developed a new fully consistent algorithm of multi-bunches simulation. We implemented two working modes, namely , `AUTO` and `FORCE`. In the first mode, only a single bunch is tracked and accelerated at the beginning, until the radial neighboring bunch effects become an important factor to the bunches' behavior. Then the new bunches will be injected automatically to take these effects into account. In this way, we can save time and memory sufficiently, and more important, we can get higher precision for the simulation in the region where neighboring bunch effects are neglectable. In the other mode, multi-bunches simulation starts from the injection points. This mode is appropriate for the machines in which this effects is unneglectable since the injection point.

In the space charge calculation for multi-bunches, the computation region covers all bunches. Because the energy of the bunches is quite different, it is inappropriate to use only one particle rest frame and a single Lorentz transformation any more. So the particles are grouped into different energy bins and in each bin the energy spread is relatively small. We apply Lorentz transforming, calculate the space charge fields and apply the back Lorentz transforming for each bin separately. Then add the field data together. Each particle has a ID number to identify which energy bin it belongs to.

5.9 Input

All the three working modes of OPAL-CYCL use an input file written in MAD language which will be described in detail in the following chapters.

For the **Tune Calculation mode**, one additional auxiliary file with the following format is needed.

72.000	2131.4	-0.240
74.000	2155.1	-0.296
76.000	2179.7	-0.319
78.000	2204.7	-0.309
80.000	2229.6	-0.264
82.000	2254.0	-0.166
84.000	2278.0	0.025

In each line the three values represent energy E , radius r and P_r for the SEO (Static Equilibrium Orbit) at starting point respectively and their units are MeV, mm, mrad.

A bash script *tuning.sh* is shown on the next page, to execute OPAL-CYCL for tune calculations.

```
#!/bin/bash
rm -rf tempfile
rm -f plotdata
rm -f tuningresult

exec 6<FIXPO_SEO
N="260"
j="1"
while read -u 6 E1 r pr
# read in Energy, initil R and intial Pr of each
# SEO from FIXPO output
do
rm -rf tempfile
echo -n j =
echo "$j"
echo -n energy= > tempfile
echo -n "$E1" >> tempfile
echo ";" >> tempfile

echo -n r= >> tempfile
echo -n "$r" >> tempfile
echo ";" >> tempfile

echo -n pr= >> tempfile
echo -n "$pr" >> tempfile
echo ";" >> tempfile
# execute OPAL to calculate tuning frquency and store
opal testcycl.in --commlib mpi --info 0 | \
grep "Max" >>tuningresult
j=$((j+1))
done
exec 6<&-
rm -rf tempfile
```

```

# post porcess
exec 8<tuningresult
  rm -f plotdata
  i="0"

while [ $i -lt $N ]
do
  read -u 8 a b url d
  read -u 8 aa bb uz1 dd
  echo "$url  $uz1" >>plotdata
  i=$((i+1))
done

exec 8<&-
rm -f tuningresult

```

To start execution, just run *tuning.sh* which uses the input file *testcycl.in* and the auxiliary file *FIXPO_SEO*. The output file is *plotdata* from which one can plot the tune diagram.

5.10 Output

Single Particle Tracking mode

The intermediate phase space data is stored in an ASCII file which can be used to plot the orbit. The file's name is combined by input file name (without extension) and *-trackOrbit.dat*. The data are stored in the global Cartesian coordinates. The frequency of the data output can be set using the option `SPTDUMPFREQ` of `OPTION` statement (see §7.3)

The phase space data per STEPPERTURN (a parameter in the `TRACK` command) steps is stored in an ASCII file. The file's name is a combination of input file name (without extension) and *-afterEachTurn.dat*. The data is stored in the global cylindrical coordinate system. Please note that if the field map is ideally isochronous, the reference particle of a given energy take exactly one revolution in STEPPERTURN steps; Otherwise, the particle may not go through a full 360° in STEPPERTURN steps.

There are 3 ASCII files which store the phase space data around 0, $\pi/8$ and $\pi/4$ azimuths. Their names are the combinations of input file name (without extension) and *-Angle0.dat*, *-Angle1.dat* and *-Angle2.dat* respectively. The data is stored in the global cylindrical coordinate system, which can be used to check the property of the closed orbit.

Tune calculation mode

The tunes ν_r and ν_z of each energy are stored in a ASCII file with name *tuningresult*.

Multi-particle tracking mode

The intermediate phase space data of all particles and some interesting parameters, including RMS envelop size, RMS emittance, external field, time, energy, length of path, number of bunches and tracking step, are stored in the H5hut file-format (<http://h5part.web.psi.ch/>) and can be analyzed using the H5root (<http://amas.web.psi.ch/tools/H5PartROOT/index.html>). The frequency of the data output can be set using the `PSDUMPFREQ` option of `OPTION` statement (see §7.3). The file is named like the input file but the extension is *.h5*.

The intermediate phase space data of central particle (with ID of 0) and an off-centering particle (with ID of 1) are stored in an ASCII file. The file's name is combined by the input file name (without extension) and *-trackOrbit.dat*. The frequency of the data output can be set using the `SPTDUMPFREQ` option of `OPTION` statement (see §7.3).

Chapter 6

Command Format

All flavours of OPAL using the same input language the MAD language. The language dialect here is ajar to MAD9, for hard core MAD8 users there is a conversion guide.

It is the first time that machines such as cyclotrons, proton and electron linacs can be described within the same language in the same simulation framework.

6.1 Statements and Comments

Input for OPAL is free format, and the line length is not limited. During reading, input lines are normally printed on the echo file, but this feature can be turned off for long input files. The input is broken up into tokens (words, numbers, delimiters etc.), which form a sequence of commands, also known as statements. Each statement must be terminated by a semicolon (;), and long statements can be continued on any number of input lines. White space, like blank lines, spaces, tabs, and newlines are ignored between tokens. Comments can be introduced with two slashes (//) and any characters following the slashes on the same line are ignored.

The C convention for comments (/ * . . . */) is also accepted. The comment delimiters / * and */ can be nested; this allows to “comment out” sections of input.

In the following descriptions, words in lower case stand for syntactic units which are to be replaced by actual text. UPPER CASE is used for keywords or names. These must be entered as shown. Ellipses (. . .) are used to indicate repetition.

The general format for a command is

```
keyword, attribute, ..., attribute;  
label: keyword, attribute, ..., attribute;
```

It has three parts:

1. The `label` is required for a definition statement. Its must be an identifier (see §6.2) and gives a name to the stored command.
2. The `keyword` identifies the action desired. It must be an identifier (see §6.2).
3. Each `attribute` is entered in one of the forms

```
attribute-name  
attribute-name=attribute-value  
attribute-name:=attribute-value
```

and serves to define data for the command, where:

- The `attribute-name` selects the attribute, it must be an identifier (see §6.2).
- The `attribute-value` gives it a value (see §6.3). When the attribute value is a constant or an expression preceded by the delimiter `=` it is evaluated immediately and the result is assigned to the attribute as a constant. When the attribute value is an expression preceded by the delimiter `:=` the expression is retained and re-evaluated whenever one of its operands changes.

Each attribute has a fixed attribute type (see §6.3).

The `attribute-value` can only be left out for logical attributes, this implies a `true` value.

When a command has a `label`, OPAL keeps the command in memory. This allows repeated execution of the same command by entering its label only:

```
label;
```

or to re-execute the command with modified attributes:

```
label, attribute, ..., attribute;
```

If the label of such a command appears together with new attributes, OPAL makes a copy of the stored command, replaces the attributes entered, and then executes the copy:

```
QF:QUADRUPOLE,L=1,K1=0.01; // first definition of QF
QF,L=2;                    // redefinition of QF

MATCH;
...
LMD:LMDIF,CALLS=10;        // first execution of LMD
LMD;                      // re-execute LMD with
                          // the same attributes
LMD,CALLS=100,TOLERANCE=1E-5; // re-execute LMD with
                          // new attributes
ENDMATCH;
```

6.2 Identifiers or Labels

An identifier refers to a keyword, an element, a beam line, a variable, an array, etc.

A label begins with a letter, followed by an arbitrary number of letters, digits, periods (`.`), underscores (`_`). Other special characters can be used in a label, but the label must then be enclosed in single or double quotes. It makes no difference which type of quotes is used, as long as the same are used at either end. The preferred form is double quotes. The use of non-numeric characters is however strongly discouraged, since it makes it difficult to subsequently process a OPAL output with another program.

When a name is not quoted, it is converted to upper case; the resulting name must be unique. An identifier can also be generated from a string expression (see §6.4).

6.3 Command Attribute Types

An object attribute is referred to by the syntax

```
object-name->attribute-name
```

If the attribute is an array (see §6.13), one of its components is found by the syntax

```
object-name->attribute-name[index]
```

The following types of command attributes are available in OPAL:

- String (see §6.4),
- Logical (see §6.5),
- Real expression (see §6.6),
- Deferred expression (see §6.8.5),
- Place (see §6.9.1),
- Range (see §6.9.2),
- Constraint (see §6.10),
- Variable Reference (see §6.11)
- Regular expression (see §6.12).
- Array (see §6.13) of
 - Logical (see §6.13.1),
 - Real (see §6.13.2),
 - String (see §6.13.3),
 - Token lists (see §6.13.4),

See also:

- Operators (see Tab. 6.4),
- Functions (see Tab. 6.5),
- Array functions (see Tab. 6.7),
- Real functions of arrays (see Tab. 6.9),
- Operand (see §6.8),
- Random generators (see §6.8.5).

6.4 String Attributes

A string attribute makes alphanumeric information available, e.g. a title, file name, element class name, or an option. It can contain any characters, enclosed in single (') or double (") quotes. However, if it contains a quote, this character must be doubled. Strings can be concatenated using the & operator (see Tab. 6.1). An operand in a string can also use the function `STRING` (see Tab. 6.2). String values can occur in string arrays (see §6.13).

Examples:

```
TITLE, "This is a title for the program run ""test""";
CALL, FILE="save";

X=1;
TWISS, LINE=LEP&STRING(X+1);
```

The second example converts the value of the expression “X+1” to a string and appends it to “LEP”, giving the string “LEP2”.

Table 6.1: String Operator in OPAL

Operator	Meaning	result type	operand types
X & Y	concatenate the strings X and Y. String concatenations are always evaluated immediately when read.	string	string,string

Table 6.2: String Function in OPAL

Function	Meaning	result type	argument type
STRING(X)	return string representation of the value of the numeric expression X	string	real

6.5 Logical Expressions

Many commands in OPAL require the setting of logical values (flags) to represent the on/off state of an option. A logical value is represented by one of the values TRUE or FALSE, or by a logical expression. A logical expression can occur in logical arrays (see §6.13.1).

A logical expression has the same format and operator precedence as a logical expression in C. It is built from logical operators (see Tab. 6.3) and logical operands:

```

relation      ::= "TRUE" |
                  "FALSE" |
                  real-expr rel-operator real-expr

rel-operator  ::= "==" | "!=" | "<" | ">" | ">=" | "<="

and-expr      ::= relation | and-expr "&&" relation

logical-expr  ::= and-expr | logical-expr "||" and-expr

```

Table 6.3: Logical Operators in OPAL

Operator	Meaning	result type	operand type
X < Y	true, if X is less than Y	logical	real,real
X <= Y	true, if X is not greater than Y	logical	real,real
X > Y	true, if X is greater than Y	logical	real,real
X >= Y	true, if X is not less than Y	logical	real,real
X == Y	true, if X is equal to Y	logical	real,real
X != Y	true, if X is not equal to Y	logical	real,real
X && Y	true, if both X and Y are true	logical	logical,logical
X Y	true, if at least one of X and Y is true	logical	logical,logical

Example:

```
OPTION,ECHO=TRUE; // output echo is desired
```

When a logical attribute is not entered, its default value is always `false`. When only its name is entered, the value is set to `TRUE`:

```
OPTION,ECHO; // same as above
```

Example of a logical expression:

```
X>10 && Y<20 || Z==15
```

6.6 Real Expressions

To facilitate the definition of interdependent quantities, any real value can be entered as an arithmetic expression. When a value used in an expression is redefined by the user or changed in a matching process, the expression is re-evaluated. Expression definitions may be entered in any order. OPAL evaluates them in the correct order before it performs any computation. At evaluation time all operands used must have values assigned. A real expression can occur in real arrays (see §6.13.2).

A real expression is built from operators (see Tab. 6.4) and operands (see §6.8):

```
real-ref  ::= real-variable |
              real-array "[" index "]" |
              object "->" real-attribute |
              object "->" real-array-attribute "[" index "]" |

table-ref ::= table "@" place "->" column-name

primary   ::= literal-constant |
              symbolic-constant |
              "#" |
              real-ref |
              table-ref |
              function-name "(" arguments ")" |
              (real-expression)

factor    ::= primary |
              factor "^" primary

term      ::= factor |
              term "*" factor |
              term "/" factor

real-expr ::= term |
              "+" term |
              "-" term |
              real-expr "+" term |
              real-expr "-" term |
```

It may contain functions (see Tab. 6.5), Parentheses indicate operator precedence if required. Constant sub-expressions are evaluated immediately, and the result is stored as a constant.

6.7 Operators

An expression can be formed using operators (see Tab. 6.4) and functions (see Tab. 6.5) acting on operands (see §6.8).

Table 6.4: Real Operators in OPAL

Operator	Meaning	result type	operand type(s)
Real operators with one operand			
+ X	unary plus, returns X	real	real
- X	unary minus, returns the negative of X	real	real
Real operators with two operands			
X + Y	add X to Y	real	real,real
X - Y	subtract Y from X	real	real,real
X * Y	multiply X by Y	real	real,real
X / Y	divide X by Y	real	real,real
X ^ Y	power, return X raised to the power Y (Y > 0)	real	real,real

Table 6.5: Real Functions in OPAL

Function	Meaning	result type	argument type(s)
Real functions with no arguments			
RANF ()	random number, uniform distribution in [0,1)	real	-
GAUSS ()	random number, Gaussian distribution with $\sigma = 1$	real	-
USER0 ()	random number, user-defined distribution	real	-

Care must be used when an ordinary expression contains a random generator. It may be re-evaluated at unpredictable times, generating a new value. However, the use of a random generator in an assignment expression is safe. Examples:

```

D:DRIFT,L=0.01*RANF();    // a drift space with rand. length,
                           // may change during execution.
P=EVAL(0.001*TGAUSS(X));  // Evaluated once
                           // and stored as a constant.

```

6.8 Operands in Expressions

A real expression may contain the operands listed in the following subsections.

6.8.1 Literal Constants

Numerical values are entered like FORTRAN constants. Real values are accepted in INTEGER or REAL format. The use of a decimal exponent, marked by the letter D or E, is permitted.

Examples:

Table 6.6: Real Functions with one in OPAL

TRUNC (X)	truncate X towards zero (discard fractional part)	real	real
ROUND (X)	round X to nearest integer	real	real
FLOOR (X)	return largest integer not greater than X	real	real
CEIL (X)	return smallest integer not less than X	real	real
SIGN (X)	return sign of X (+1 for X positive, -1 for X negative, 0 for X zero)	real	real
SQRT (X)	return square root of X	real	real
LOG (X)	return natural logarithm of X	real	real
EXP (X)	return exponential to the base e of X	real	real
SIN (X)	return trigonometric sine of X	real	real
COS (X)	return trigonometric cosine of X	real	real
ABS (X)	return absolute value of X	real	real
TAN (X)	return trigonometric tangent of X	real	real
ASIN (X)	return inverse trigonometric sine of X	real	real
ACOS (X)	return inverse trigonometric cosine of X	real	real
ATAN (X)	return inverse trigonometric tangent of X	real	real
TGAUSS (X)	random number, Gaussian distribution with $\sigma=1$, truncated at X	real	real
USER1 (X)	random number, user-defined distribution with one parameter	real	real
EVAL (X)	evaluate the argument immediately and transmit it as a constant	real	real
Real functions with two arguments			
ATAN2 (X, Y)	return inverse trigonometric tangent of Y/X	real	real,real
MAX (X, Y)	return the larger of X, Y	real	real,real
MIN (X, Y)	return the smaller of X, Y	real	real,real
MOD (X, Y)	return the largest value less than Y which differs from X by a multiple of Y	real	real,real
USER2 (X, Y)	random number, user-defined distribution with two parameters	real	real,real

1, 10.35, 5E3, 314.1592E-2

6.8.2 Symbolic constants

OPAL recognises a few built-in mathematical and physical constants (see Tab. 6.8). Their names must not be used for user-defined labels. Additional symbolic constants may be defined (see §7.4.2) to simplify their repeated use in statements and expressions.

Table 6.7: Real Functions of Arrays in OPAL

Function	Meaning	result type	operand type
VMAX (<i>X</i> , <i>Y</i>)	return largest array component	real	real array
VMIN (<i>X</i> , <i>Y</i>)	return smallest array component	real	real array
VRMS (<i>X</i> , <i>Y</i>)	return rms value of an array	real	real array
VABSMAX (<i>X</i> , <i>Y</i>)	return absolute largest array component	real	real array

Table 6.8: Predefined Symbolic Constants

OPAL name	Mathematical symbol	Value	Unit
PI	π	3.1415926535898	1
TWOPI	2π	6.2831853071796	1
RADDEG	$180/\pi$	57.295779513082	rad/deg
DEGRAD	$\pi/180$.017453292519943	deg/rad
E	e	2.7182818284590	1
EMASS	m_e	.51099906e-3	GeV
PMASS	m_p	.93827231	GeV
HMASS	m_{h^-}	.939277	GeV
CMASS	m_c	12*0.931494027	GeV
UMASS	m_u	238*0.931494027	GeV
MMASS	m_μ	0.1057	GeV
DMASS	m_d	2*0.931494027	GeV
XEMASS	m_{xe}	124*0.931494027	GeV
CLIGHT	c	299792458	m/s
OPALVERSION		120	for 1.2.0

6.8.3 Variable labels

Often a set of numerical values depends on a common variable parameter. Such a variable must be defined as a global variable (see §7.4.1) defined by one of

```

X=expression;
X:=expression;
VECTOR X=vector-expression;
VECTOR X:=vector-expression;

```

When such a variable is used in an expression, OPAL uses the current value of the variable. When the value is a constant or an expression preceded by the delimiter = it is evaluated immediately and the result is assigned to the variable as a constant. When the value is an expression preceded by the delimiter := the expression is retained and re-evaluated whenever one of its operands changes. Example:

```
L=1.0;
X:=L;
D1:DRIFT,L:=X;
D2:DRIFT,L:=2.0-X;
```

When the value of X is changed, the lengths of the drift spaces are recalculated as X and 2-X respectively.

6.8.4 Element or command attributes

In arithmetic expressions the attributes of physical elements or commands can occur as operands. They are named respectively by

```
element-name->attribute-name
command-name->attribute-name
```

If they are arrays, they are denoted by

```
element-name->attribute-name[index]
command-name->attribute-name[index]
```

Values are assigned to attributes in element definitions or commands.

Example:



```
D1:DRIFT,L=1.0;
D2:DRIFT,L=2.0-D1->L;
```

D1->L refers to the length L of the drift space D1.

6.8.5 Deferred Expressions and Random Values

Definition of random machine imperfections requires evaluation of expressions containing random functions. These are not evaluated like other expressions before a command begins execution, but sampled as required from the distributions indicated when errors are generated. Such an expression is known as a **deferred expression**. Its value cannot occur as an operand in another expression.

6.9 Element Selection

Many OPAL commands allow for the possibility to process or display a subset of the elements occurring in a beam line or sequence. This is not yet available in:  OPAL-T and  OPAL-CYCL.

6.9.1 Element Selection

A place denotes a single element, or the position **following** that element. It can be specified by one of the choices

object-name[index] The name *object-name* is the name of an element, line or sequence, and the integer *index* is its occurrence count in the beam line. If the element is unique, [*index*] can be omitted.

#S denotes the position before the first physical element in the **full** beam line. This position can also be written #0.

#E denotes the position after the last physical element in the **full** beam line.

Either form may be qualified by one or more beam line names, as described by the formal syntax:

```
place ::= element-name |
         element-name "[" integer "]" |
         "#S" |
         "#E" |
         line-name ":" place
```

An omitted index defaults to one. Examples: assume the following definitions:

```
M: MARKER;
S: LINE= (C, M, D) ;
L: LINE= (A, M, B, 2*S, A, M, B) ;
SURVEY, LINE=L
```

The line L is equivalent to the sequence of elements

```
A, M, B, C, M, D, C, M, D, A, M, B
```

Some possible `place` definitions are:

C[1] The first occurrence of element C.

#S The beginning of the line L.

M[2] The second marker M at top level of line L, i. e. the marker between second A and the second B.

#E The end of line L

S[1]::M[1] The marker M nested in the first occurrence of S.

6.9.2 Range Selection

A range in a beam line (see §9) is selected by the following syntax:

```
range ::= place |
         place "/" place
```

This denotes the range of elements from the first `place` to the second `place`. Both positions are included. A few special cases are worth noting:

- When `place1` refers to a `LINE` (see §9). the range starts at the **beginning** of this line.
- When `place2` refers to a `LINE` (see §9). the range ends at the **ending** of this line.
- When both `place` specifications refer to the same object, then the second can be omitted. In this case, and if `place` refers to a `LINE` (see §9) the range contains the whole of the line.

Examples: Assume the following definitions:

```
M: MARKER;
S: LINE= (C, M, D) ;
L: LINE= (A, M, B, 2*S, A, M, B) ;
```

The line L is equivalent to the sequence of elements

```
A, M, B, C, M, D, C, M, D, A, M, B
```

Examples for range selections:



#S/#E The full range or L.

A[1]/A[2] A[1] through A[2], both included.

S::M/S[2]::M From the marker M nested in the first occurrence of S, to the marker M nested in the second occurrence of S.

S[1]/S[2] Entrance of first occurrence of S through exit of second occurrence of S.

6.10 Constraints

Please note this is not yet available in:  OPAL-T and  OPAL-CYCL.

In matching it is desired to specify equality constraints, as well as lower and upper limits for a quantity. OPAL accepts the following form of constraints:

```
constraint          ::= array-expr constraint-operator array-expr
constraint-operator ::= "==" | "<" | ">"
```

6.11 Variable Names

A variable name can have one of the formats:

```
variable name ::= real variable |
                object->"real attribute"
```

The first format refers to the value of the `global variable` (see §7.4.1), the second format refers to a named `attribute` of the named `object`. `object` can refer to an element or a command

6.12 Regular Expressions

Some commands allow selection of items via a `regular-expression`. Such a pattern string **must** be enclosed in single or double quotes; and the case of letters is significant. The meaning of special characters follows the standard UNIX usage: utility:

. Stands for a single arbitrary character,

[letter...letter] Stands for a single character occurring in the bracketed string, Example: “[abc]” denotes the choice of one of a, b, c.

[character-character] Stands for a single character from a range of characters, Example: “[a-zA-Z]” denotes the choice of any letter.

***** Allows zero or more repetitions of the preceding item, Example: “[A-Z]*” denotes a string of zero or more upper case letters.

\character Removes the special meaning of `character`, Example: “*” denotes a literal asterisk.

All other characters stand for themselves. The pattern

```
"[A-Za-z][A-Za-z0-9_']*"
```

illustrates all possible unquoted identifier formats (see §6.2). Since identifiers are converted to lower case, after reading they will match the pattern

```
"[a-z][a-z0-9_']*"
```

Examples for pattern use:

```
SELECT,PATTERN="D.."
SAVE,PATTERN="K.*QD.*\R1"
```

The first command selects all elements whose names have exactly three characters and begin with the letter `D`. The second command saves definitions beginning with the letter `K`, containing the string `QD`, and ending with the string `.R1`. The two occurrences of `.*` each stand for an arbitrary number (including zero) of any character, and the occurrence `\.` stands for a literal period.

6.13 Arrays

An attribute array is a set of values of the same `attribute type` (see §6.3). Normally an array is entered as a list in braces:

```
{value,...,value}
```

The list length is only limited by the available storage. If the array has only one value, the braces `()` can be omitted:

```
value
```

6.13.1 Logical Arrays

For the time being, logical arrays can only be given as a list. The formal syntax is:

```
logical-array ::= "{" logical-list "}"

logical-list  ::= logical-expr |
                  logical-list "," logical-expr
```

Example:

```
{true,true,a==b,false,x>y && y>z,true,false}
```

6.13.2 Real Arrays

Real arrays have the following syntax:

array-ref	::= array-variable object ">" array-attribute
table-ref	::= "ROW" "(" table "," place ")" "ROW" "(" table "," place "," column-list ")" "COLUMN" "(" table "," column ")" "COLUMN" "(" table "," column "," range ")"
columns	::= column "{" column-list "}"
column-list	::= column column-list "," column
column	::= string
real-list	::= real-expr real-list "," real-expr
index-select	::= integer integer "," integer integer "," integer "," integer
array-primary	::= "{" real-list "}" "TABLE" "(" index-select "," real-expr ")" array-ref table-ref array-function-name "(" arguments ")" (array-expression)
array-factor	::= array-primary array-factor "^" array-primary
array-term	::= array-factor array-term "*" array-factor array-term "/" array-factor
array-expr	::= array-term "+" array-term "-" array-term array-expr "+" array-term array-expr "-" array-term

Example:

```
{a,a+b,a+2*b}
```

There are also three functions allowing the generation of real arrays:

TABLE Generate an array of expressions:

```
TABLE(n2,expression)    // implies
                        // TABLE(1:n2:1,expression)
TABLE(n1:n2,expression) // implies
                        // TABLE(n1:n2:1,expression)
TABLE(n1:n2:n3,expression)
```

These expressions all generate an array with $n2$ components. The components selected by $n1:n2:n3$ are filled from the given expression; a C pseudo-code for filling is

Table 6.9: Real Array Functions in OPAL (acting component-wise)

Function	Meaning	result type	argument type
TRUNC (X)	truncate X towards zero (discard fractional part)	real array	real array
ROUND (X)	round X to nearest integer	real array	real array
FLOOR (X)	return largest integer not greater than X	real array	real array
CEIL (X)	return smallest integer not less than X	real array	real array
SIGN (X)	return sign of X (+1 for X positive, -1 for X negative, 0 for X zero)	real array	real array
SQRT (X)	return square root of X	real array	real array
LOG (X)	return natural logarithm of X	real array	real array
EXP (X)	return exponential to the base e of X	real array	real array
SIN (X)	return trigonometric sine of X	real array	real array
COS (X)	return trigonometric cosine of X	real array	real array
ABS (X)	return absolute value of X	real array	real array
TAN (X)	return trigonometric tangent of X	real array	real array
ASIN (X)	return inverse trigonometric sine of X	real array	real array
ACOS (X)	return inverse trigonometric cosine of X	real array	real array
ATAN (X)	return inverse trigonometric tangent of X	real array	real array
TGAUSS (X)	random number, Gaussian distribution with $\sigma=1$, truncated at X	real array	real array
USER1 (X)	random number, user-defined distribution with one parameter	real array	real array
EVAL (X)	evaluate the argument immediately and transmit it as a constant	real array	real array

```
int i;
for (i = n1; i <= n2; i += n3) a[i] = expression(i);
```

In each generated expression the special character hash sign (#) is replaced by the current value of the index i .

Example:

```
// an array with 9 components, evaluates to
// {1,4,7,10,13}:
table(5:9:2,3*#+1) // equivalent to
// {0,0,0,0,16,0,22,0,28}
```

ROW Generate a table row:

```
ROW(table,place)           // implies all columns
ROW(table,place,column list)
```

This generates an array containing the named (or all) columns in the selected place.

COLUMN Generate a table column:

```
COLUMN(table,column)           // implies all rows  
COLUMN(table,column,range)
```

This generates an array containing the selected (or all) rows of the named column.

6.13.3 String Arrays

String arrays can only be given as lists of single values. For permissible values String values (see §6.4).

Example:

```
{A, "xyz", A & STRING(X) }
```

6.13.4 Token List Arrays

Token list arrays are always lists of single token lists.

Example:

```
{X:12:8,Y:12:8}
```

Chapter 7

Control Statements

7.1 Getting Help

7.1.1 HELP Command

A user who is uncertain about the attributes of a command should try the command `HELP`, which has three formats:

```
HELP;                // Give help on the "HELP" command
HELP,NAME=label;     // List funct. and attr. types of
                    // "label"
HELP,label;          // Shortcut for the second format
```

`label` is an identifier (see §6.2). If it is non-blank, OPAL prints the function of the object `label` and lists its attribute types. Entering `HELP` alone displays help on the `HELP` command.

Examples:

```
HELP;
HELP,NAME=TWISS;
HELP,TWISS;
```

7.1.2 SHOW Command

The `SHOW` statement displays the current attribute values of an object. It has three formats:

```
SHOW;                // Give help on the "SHOW" command
SHOW,NAME=pattern;   // Show names matching of "pattern"
SHOW,pattern;        // Shortcut for the second format
```

`pattern` is a regular expression (see §6.12). If it is non-blank, OPAL displays all object names matching the pattern. Entering `SHOW` alone displays help on the `SHOW` command. Examples:

```
SHOW;
SHOW,NAME="QD.*\..L*";
SHOW,"QD.*\..L*";
```

7.1.3 WHAT Command

The `WHAT` statement displays all object names matching a given regular expression. It has three formats:

```
WHAT;                // Give help on the "WHAT" command
WHAT,NAME=label;     // Show definition of "label"
WHAT,label;          // Shortcut for the second format
```

label is an identifier (see §6.2). If it is non-blank, OPAL displays the object label in a format similar to the input statement that created the object. Entering WHAT alone displays help on the WHAT command. Examples:

```
WHAT;
WHAT,NAME=QD;
WHAT,QD;
```

7.2 STOP / QUIT Statement

The statement

```
STOP or QUIT;
```

terminates execution of the OPAL program, or, when the statement occurs in a CALL file (see §7.7.1), returns to the calling file. Any statement following the STOP or QUIT statement is ignored.

7.3 OPTION Statement

The OPTION command controls global command execution and sets a few global quantities:

```
OPTION,ECHO=logical,INFO=logical,TRACE=logical,
      VERIFY=logical,WARN=logical,
      SEED=real,TELL=logical,PSDUMPFREQ=integral,
      STATDUMPFREQ=integral,SPTDUMPFREQ=integral,
      REPARTFREQ=integral,REBINFREQ=integral;
```

The first five logical flags activate or deactivate execution options:

ECHO Controls printing of an echo of input lines on the standard error file.

INFO If this option is turned off, OPAL suppresses all information messages. It also affects the *gnu.out* and *eb.out* files in case of OPAL-CYCL simulations.

TRACE When the TRACE option is on, OPAL writes additional trace information on the standard error file for each executable command. This information includes the command name and elapsed CPU time before and after the command.

VERIFY If this option is on, OPAL gives a message for each undefined variable or element in a beam line.

WARN If this option is turned off, OPAL suppresses all warning messages.

SEED Selects a particular sequence of random values. A SEED value is an integer in the range [0...999999999] (default: 123456789). SEED can be an expression. If SEED = -1, the time is used as seed and the generator is not portable anymore. See also: random values (see §6.8.5).

PSDUMPFREQ Defines after how many time steps the phase space is dumped into the H5hut file. Default value is 10.

STATDUMPFREQ Defines after how many time steps we dump statistical data, such as RMS beam emittance, to the .stat file. The default value is 10. Currently only available for OPAL-T.

SPTDUMPFREQ Defines after how many time steps we dump the phase space of single particle. It is always useful to record the trajectory of reference particle or some specified particle for primary study. Its default value is 1.

REPARTFREQ Defines after how many time steps we do particles repartition to balance the computational load of the computer nodes. Its default value is 10.

REBINFREQ Defines after how many time steps we update the energy Bin ID of each particle. For the time being. Only available for multi-bunch simulation in OPAL-CYCL. Its default value is 100.

PSDUMPEACHTURN Control option of phase space dumping. If true, dump phase space after each turn. For the time being, this is only use for multi-bunch simulation in OPAL-CYCL. Its default set is false.

PSDUMPLOCALFRAME Control option whether the phase space data is dumped in the global Cartesian frame or in the local Cartesian frame. If true, in local frame, otherwise in global Cartesian frame. Only available for OPAL-CYCL. Its default set is false. Note that restarting run cannot be launched by reading in phase space data in local frame.

SCSOLVEFREQ If the space charge field is slowly varying w.r.t. external fields, this option allows to change the frequency of space charge calculation, i.e. the space charge forces are evaluated every SCSOLVEFREQ step and then reused for the following steps. Affects integrators LF-2 and RK-4 of OPAL-CYCL. Its default value is 1. Note: as the multiple-time-stepping (MTS) integrator maintains accuracy much better with reduced space charge solve frequency, this option should probably not be used anymore.

MTSSUBSTEPS Only used for multiple-time-stepping (MTS) integrator in OPAL-CYCL. Specifies how many substeps for external field integration are done per step. Default value is 1. Making less steps per turn and increasing this value is the recommended way to reduce space charge solve frequency.

RHODUMP If true the scalar ρ field is saved each time a phase space is written. There exists a reader in Visit with versions greater or equal 1.11.1.

EFDUMP Not supported anymore.

EBDUMP If true the electric and magnetic field on the particle is saved each time a phase space is written.

CSRDUMP If true the electric csr field component (E_z), line density and the derivative of the line density is written into the *data* directory.

AUTOPHASE A phase scan of all n rf-elements is performed, if AUTOPHASE is greater than zero. AUTOPHASE finds the maximum energy in a way similar to the code Astra.

1. find the phase ϕ_i for maximum energy of the i -th cavity.
2. track is continued with $\phi_i + LAG$ to the element $i + 1$.
3. if $i < n$ goto 1

For convenience a file (`inputfn.phases`) with the phases corresponding to the maximum energies is written. A AUTOPHASE value of 4 gives Astra comparable results. An example is given in (see §3.2).

SCAN If true one can simulate in a loop several machines where some variables can be random variables. Find an example at 15.1.1.

CZERO If true the distributions are generated such that the centroid is exactly zero and not statistically dependent.

RNGTYPE The default random number generator (RANDOM) is a portable 48-bit generator. Three quasi random generators are available:

1. HALTON
2. SOBOL
3. NIEDERREITER.

For details see the GSL reference manual (18.5).

ENABLEHDF5 If true (default), HDF5 read and write is enabled.

ASCIIDUMP If true, instead of HDF5, ASCII output is generated for the following elements: Probe, Collimator, Monitor, Stripper, Foil and global losses.

NLHS Number of stored old solutions for extrapolating the new starting vector. Default value is 1 and just the last solution is used.

NUMBLOCKS Maximum number of vectors in the Krylov space (for RCGSolMgr). Default value is 0 and BlockCGSolMgr will be used.

RECYCLEBLOCKS Number of vectors in the recycle space (for RCGSolMgr). Default value is 0 and BlockCGSolMgr will be used.

SCHOTTKYCORR Not used in this version of OPAL.

SCHOTTKYRENO Not used in this version of OPAL.

BOUNDPDESTROYFQ The frequency to do `boundp_destroy` to delete lost particles. Default 10

REMOTEPARTDEL Artificially delete the remote particle if its distance to the beam mass is larger than REMOTEPARTDEL times of the beam rms size, its default values is -1 (no delete)

The last attribute requests listing of the current settings:

TELL If true, the current settings are listed.

Examples:

```
OPTION, ECHO=FALSE, TELL;
OPTION, SEED=987456321
```

7.4 Parameter Statements

7.4.1 Variable Definitions

OPAL recognises several types of variables.

Table 7.1: Default Settings for Options

ECHO	= false	INFO	= true	TRACE	= false
WARN	= true	VERIFY	= false	SEED	= 123456789
PSDUMPFREQ	= 10	SPTDUMPFREQ	= 1	REPARTFREQ	= 10
RHODUMP	= false	CZERO	= false	TELL	= false
VERIFY	= false	STATDUMPFREQ	= 10	RNGTYPE	= RANDOM
EBDUMP	= false	SCSOLVEFREQ	= 1	REBINFREQ	= 100
SCAN	= false	AUTOPHASE	= 0	PPDEBUG	= false
SURFDUMPFREQ	= -1	PSDUMPEACHTURN	= false	PSDUMLOCALFRAME	= false
CSRDUMP	= false	ENABLEHDF5	= true	ASCIIDUMP	= false
REMPARTDEL	= -1	NUMBLOCKS	= true	RECYCLEBLOCKS	= false
NLHS	= -1	BOUNDPDESTROYFQ	= false	SCHOTTKYCORR	= 0.0
SCHOTTKYRENO	= -1		=		=

Real Scalar Variables

```
REAL variable-name=real-expression;
```

The keyword REAL is optional. For backward compatibility the program also accepts the form

```
variable-name:=real-expression;
```

This statement creates a new global variable `variable-name` and discards any old variable with the same name. Its value depends on all quantities occurring in `real-expression` (see §6.6). Whenever an operand changes in `real-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

An assignment in the sense of the FORTRAN or C languages can be achieved by using the EVAL function (see §7.4.4).

A reserved variable is the value P0 which is used as the global reference momentum for normalising all magnetic field coefficients. Example:

```
REAL GEV=100;
P0=GEV;
```

Circular definitions are not allowed:

```
X=X+1;    // X cannot be equal to X+1
A=B;
B=A;      // A and B are equal, but of unknown value
```

However, redefinitions by assignment are allowed:

```
X=EVAL (X+1) ;
```

Real Vector Variables

```
REAL VECTOR variable-name=vector-expression;
```

The keyword `REAL` is optional. This statement creates a new global variable `variable-name` and discards any old variable with the same name. Its value depends on all quantities occurring in `vector-expression` (see §6.13) on the right-hand side. Whenever an operand changes in `vector-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

Example:

```
REAL VECTOR A = TABLE(10, #);
REAL VECTOR B = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

Circular definitions are not allowed, but redefinitions by assignment are allowed.

Logical Variables

```
BOOL variable-name=logica-expression;
```

This statement creates a new global variable `variable-name` and discards any old variable with the same name. Its value depends on all quantities occurring in `logical-expression` (see §6.5). Whenever an operand changes in `logical-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

Example:

```
BOOL FLAG = X != 0;
```

Circular definitions are not allowed, but redefinitions by assignment are allowed.

7.4.2 Symbolic Constants

OPAL recognises a few built-in mathematical and physical constants. (see Tab. 6.8) Additional constants can be defined by the command

```
REAL CONST label:CONSTANT=<real-expression>;
```

which defines a constant with the name `label`. The keyword `REAL` is optional, and `label` must be unique. An existing symbolic constant can never be redefined. The `real-expression` is evaluated at the time the `CONST` definition is read, and the result is stored as the value of the constant.

Example:

```
CONST IN=0.0254; // conversion of inches to metres
```

7.4.3 Vector Values

A vector of expressions is established by a statement

```
REAL VECTOR vector-name=vector-expression;
```

The keyword `REAL` is optional. It creates a new global vector `vector-name` and discards any old vector with the same name. Its value depends on all quantities occurring in `vector-expression` (see §6.13). Whenever an operand changes in `vector-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

Example:

```
VECTOR A_AMPL={2.5e-3,3.4e-2,0,4.5e-8};
VECTOR A_ON=TABLE(10,1);
```

Circular definitions are not allowed.

7.4.4 Assignment to Variables

A value is assigned to a variable or vector by using the function `EVAL(real-expression)`. When seen, this function is immediately evaluated and replaced by the result treated like a constant.

```
variable-name=EVAL(real-expression);
```

This statement acts like a FORTRAN or C assignment. The `real-expression` or `vector-expression` is **evaluated**, and the result is assigned as a constant to the variable or vector on the left-hand side. Finally the expression is discarded. The `EVAL` function can also be used within an expression, e. g.:

```
vector-name=TABLE(range,EVAL(real-expression));
vector-name={...,EVAL(real-expression),...};
```

A sequence like the following is permitted:

```
...                // some definitions
X=0;                // create variable X with value
                    // zero
WHILE (X <= 0.10) {
  TWISS,LINE=...;    // uses X=0, 0.01, ..., 0.10
  X=EVAL(X+.01);     // increment variable X by 0.01
                    // CANNOT use: X=X+.01;
}
```

7.4.5 VALUE: Output of Expressions

The statement

```
VALUE,VALUE=expression-vector;
```

evaluates a set of expressions using the most recent values of any operands and prints the results on the standard error file.

Example:

```
A=4;
VALUE, VALUE=TABLE (5, #*A) ;
P1=5;
P2=7;
VALUE, VALUE={P1, P2, P1*P2-3};
```

These commands give the results:

```
value: {0*A, 1*A, 2*A, 3*A, 4*A} = {0, 4, 8, 12, 16}
value: {P1, P2, P1*P2-3} = {5, 7, 32}
```

This commands serves mainly for printing one or more quantities which depend on matched attributes. It also allows use of OPAL as a programmable calculator. One may also tabulate functions.

7.5 Miscellaneous Commands

7.5.1 ECHO Statement

The ECHO statement has two formats:

```
ECHO, MESSAGE=message;
ECHO, message;           // shortcut
```

message is a string value (see §6.4). It is immediately transmitted to the ECHO stream.

7.5.2 SYSTEM: Execute System Command

During an interactive OPAL session the command SYSTEM allows to execute operating system commands. After execution of the system command, successful or not, control returns to OPAL. At present this command is only available under UNIX or VM/CMS. It has two formats:

```
SYSTEM, CMD=string;
SYSTEM, string;           // shortcut
```

The string (see §6.4) string must be a valid operating system command.

7.5.3 SYSTEM Command under UNIX

Most UNIX commands can be issued directly.

Example:

```
SYSTEM, "ls -l"
```

causes a listing of the current directory in long form on the terminal.

7.6 TITLE Statement

The TITLE statement has three formats:

```
TITLE,STRING=page-header;    // define new page header
TITLE,page-header;          // shortcut for first format
TITLE,STRING="";            // clear page header
```

`page-header` is a string value (see §6.4). It defines the page header which will be used as a title for subsequent output pages. Before the first `TITLE` statement is encountered, the page header is empty. It can be redefined at any time.

7.7 File Handling

7.7.1 CALL Statement

The `CALL` command has two formats:

```
CALL,FILE=file-name;
CALL,file-name;
```

`file-name` is a string (see §6.4). The statement causes the input to switch to the named file. Input continues on that file until a `STOP` or an end of file is encountered. Example:

```
CALL,FILE="structure";
CALL,"structure";
```

7.7.2 SAVE Statement

The `SAVE` command has two formats:

```
SAVE,FILE=file-name
```

`file-name` is a string (see §6.4). The command causes all beam element, beam line, and parameter definitions to be written on the named file. Examples:

```
SAVE,FILE="structure";
SAVE,"structure";
```

7.8 IF: Conditional Execution

Conditional execution can be requested by an `IF` statement. It allows usages similar to the C language `if` statement:

```
IF (logical) statement;
IF (logical) statement; ELSE statement;
IF (logical) { statement-group; }
IF (logical) { statement-group; }
    ELSE { statement-group; }
```

Note that all statements must be terminated with semicolons (;), but there is no semicolon after a closing brace. The statement or group of statements following the `IF` is executed if the condition is satisfied. If the condition is false, and there is an `ELSE`, the statement or group following the `ELSE` is executed.

7.9 WHILE: Repeated Execution

Repeated execution can be requested by a `WHILE` statement. It allows usages similar to the C language `while` statement:

```
WHILE (logical) statement;
WHILE (logical) { statement-group; }
```

Note that all statements must be terminated with semicolons (;), but there is no semicolon after a closing brace. The condition is re-evaluated in each iteration. The statement or group of statements following the `WHILE` is repeated as long as the condition is satisfied. Of course some variable(s) must be changed within the `WHILE` group to allow the loop to terminate.

7.10 MACRO: Macro Statements (Subroutines)

Subroutine-like commands can be defined by a `MACRO` statement. It allows usages similar to C language function call statements. A macro is defined by one of the following statements:

```
name(formals): MACRO { token-list }
name(): MACRO { token-list }
```

A macro may have formal arguments, which will be replaced by actual arguments at execution time. An empty formals list is denoted by (). Otherwise the `formals` consist of one or more names, separated by commas. The `token-list` consists of input tokens (strings, names, numbers, delimiters etc.) and is stored unchanged in the definition.

A macro is executed by one of the statements:

```
name(actuals);
name();
```

Each actual consists of a set of tokens which replaces all occurrences of the corresponding formal name. The actuals are separated by commas. Example:

```
// macro definitions:
SHOWIT(X): MACRO {
    SHOW, NAME = X;
}
DOIT(): MACRO {
    DYNAMIC, LINE=RING, FILE="DYNAMIC.OUT";
}

// macro calls:
SHOWIT(PI);
DOIT();
```

Chapter 8

Elements

8.1 Element Input Format

All physical elements are defined by statements of the form

```
label:keyword, attribute,..., attribute
```

where

label

Is the name to be given to the element (in the example QF), it is an `identifier` (see §6.2).

keyword

Is a `keyword` (see §6.2), it is an element type keyword (in the example QUADRUPOLE),

attribute

normally has the form

```
attribute-name=attribute-value
```

attribute-name

selects the attribute from the list defined for the element type `keyword` (in the example L and K1). It must be an `identifier` (see §6.2)

attribute-value

gives it a `value` (see §6.3) (in the example 1.8 and 0.015832).

Omitted attributes are assigned a default value, normally zero.

Example:

```
QF: QUADRUPOLE, L=1.8, K1=0.015832;
```

8.2 Common Attributes for all Elements

The following attributes are allowed on all elements:

TYPE A `string value` (see §6.4). It specifies an “engineering type” and can be used for element selection.

APERTURE A real vector with an arbitrary length which describes the element aperture. It is ignored by OPAL, but it can be used in other programs.

WAKEF Defines the type of wake to be applied: *WT*, *WL* or *WTL* for transverse, longitudinal or both.

other All elements can have arbitrary additional attributes which are not defined below. Such attributes must have a name different from all defined attributes and single real values.

Only the *TYPE* attribute is used by OPAL, but the *SAVE* command (see §7.7.2) saves all attributes.

8.3 Drift Spaces


```
label:DRIFT, TYPE=string, APERTURE=real-vector, L=real;
```

A DRIFT space has one real attribute:

L The drift length (default: 0 m)

Examples:

```
DR1:DRIFT, L=1.5;
DR2:DRIFT, L=DR1->L, TYPE=DRF;
```

The length of DR2 will always be equal to the length of DR1. The reference system for a drift space is a Cartesian coordinate system. This is a restricted feature:  OPAL-CYCL . In OPAL-T drifts are implicitly given, if no field is present.

8.4 Bending Magnets

Bending magnets refer to dipole fields that bend particle trajectories. Currently OPAL supports three different bend elements: RBEND (valid in OPAL-T, see 8.4.1), SBEND (valid in OPAL-T, see 8.4.2) and SBEND3D (valid in OPAL-CYCL, see 8.4.6).

Describing a bending magnet can be somewhat complicated as there can be many parameters to consider: bend angle, bend radius, entrance and exit angles etc. Therefore we have divided this section into several parts:

1. Section 8.4.6 is self contained. It describes how to implement an SBEND3D element in an OPAL-CYCL simulation.
2. Sections 8.4.1 and 8.4.2 describe the geometry and attributes of the OPAL-T bend elements RBEND and SBEND.
3. Section 8.4.3 describes how to implement an RBEND or SBEND in an OPAL-T simulation.

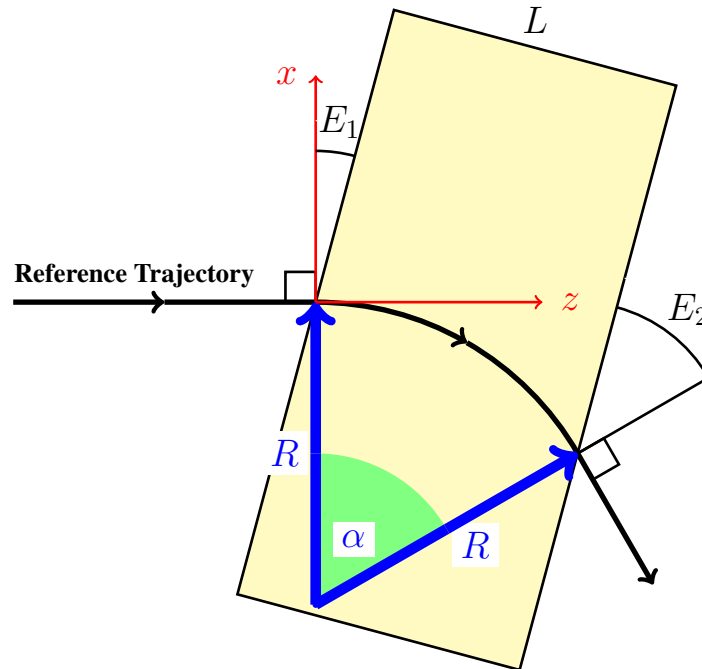


Figure 8.1: Illustration of a general rectangular bend (RBEND) with a positive bend angle α . The entrance edge angle, E_1 , is positive in this example. An RBEND has parallel entrance and exit pole faces, so the exit angle, E_2 , is uniquely determined by the bend angle, α , and E_1 ($E_2 = \alpha - E_1$). For a positively charge particle, the magnetic field is directed out of the page.

8.4.1 RBend (OPAL-T)

An RBEND is a rectangular bending magnet. The key property of an RBEND is that it has parallel pole faces. Figure 8.1 shows an RBEND with a positive bend angle and a positive entrance edge angle.

ELEMEDGE The edge of the bend is specified in s coordinates in meters. This edge corresponds to the origin of the (x,z) axes shown in Figure 8.1 and is the physical start of the bend. (Note that in general the bend fringe

fields will extend in front of this position.) The physical end of the magnet is determined by `ELEMEDGE` and the physical length of the magnet. (Note again that in general the bend fringe fields will extend past the physical end of the bend.)

L Physical length of magnet (meters, see Figure 8.1).

GAP Full vertical gap of the magnet (meters).

HAPERT Non-bend plane aperture of the magnet (meters). (Defaults to one half the bend radius.)

ANGLE Bend angle (radians). Field amplitude of bend will be adjusted to achieve this angle. (Note that for an `RBEND`, the bend angle must be less than $\pi + E1$, where $E1$ is the entrance edge angle. See below.)

K0 Field amplitude in y direction (Tesla). If the `ANGLE` attribute is set, `K0` is ignored.

K0S Field amplitude in x direction (Tesla). If the `ANGLE` attribute is set, `K0S` is ignored.

K1 Field gradient index of the magnet, $K_1 = -\frac{R}{B_y} \frac{\partial B_y}{\partial x}$, where R is the bend radius as defined in Figure 8.1.

E1 Entrance edge angle (radians). Figure 8.1 shows the definition of a positive entrance edge angle. (Note that the exit edge angle is fixed in an `RBEND` element to $E2 = ANGLE - E1$).

ROTATION Rotation of magnet about its z axis (radians). Setting this angle allows us to achieve a bend in any direction in the x/y plane.

BETA Rotation of magnet about its x axis (radians). See Figure 8.2.

DESIGNENERGY Energy of the bend reference particle (eV). The reference particle travels the path shown in Figure 8.1.

FMAPFN Name of the field map for the magnet. Currently maps in the `T7` format of type `1DProfile1` can be used (see C.11). The default option for this attribute is `FMAPN = "1DPROFILE1-DEFAULT"` (see 8.4.5). The field map is used to describe the fringe fields of the magnet (see ??).

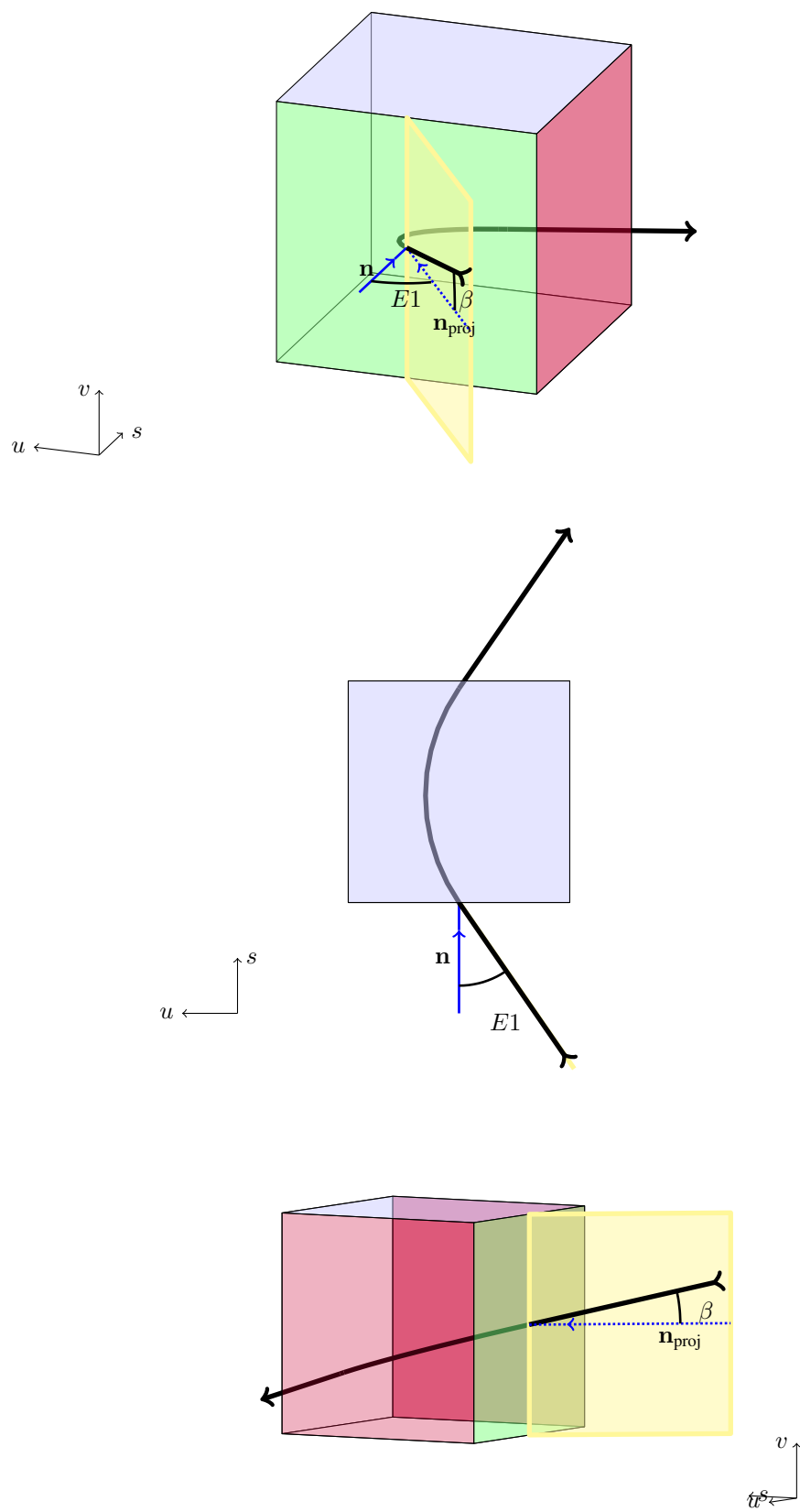


Figure 8.2: Visualisation of angles used to rotate the bend relative to the incoming beam, where \mathbf{n} is the normal of the face.

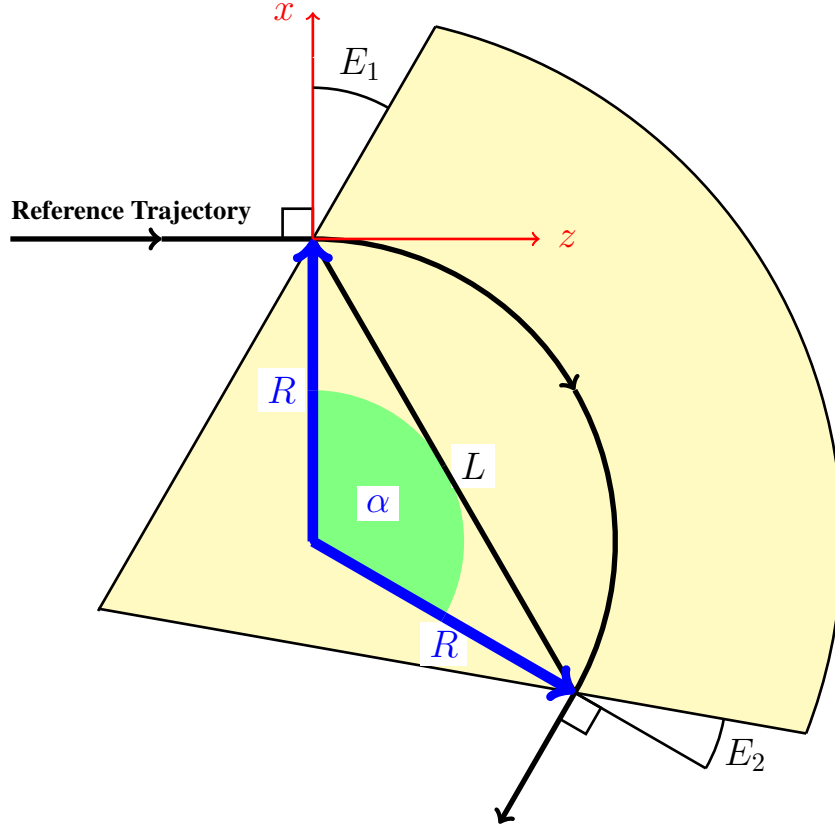


Figure 8.3: Illustration of a general sector bend (SBEND) with a positive bend angle α . In this example the entrance and exit edge angles E_1 and E_2 have positive values. For a positively charge particle, the magnetic field is directed out of the page.

8.4.2 SBend (OPAL-T)

An SBEND is a sector bending magnet. An SBEND can have independent entrance and exit edge angles. Figure 8.3 shows an SBEND with a positive bend angle, a positive entrance edge angle, and a positive exit edge angle.

ELEMEDGE The edge of the bend is specified in s coordinates in meters. This edge corresponds to the origin of the (x,z) axes shown in Figure 8.1 and is the physical start of the bend. (Note that in general the bend fringe fields will extend in front of this position.) The physical end of the magnet is determined by ELEMEDGE, the chord length of the magnet and the bend angle. (Note again that in general the bend fringe fields will extend past the physical end of the bend.)

L Chord length of the bend reference arc in meters (see Figure 8.3), given by:

$$L = 2R \sin\left(\frac{\alpha}{2}\right)$$

GAP Full vertical gap of the magnet (meters).

HAPERT Non-bend plane aperture of the magnet (meters). (Defaults to one half the bend radius.)

ANGLE Bend angle (radians). Field amplitude of bend will be adjusted to achieve this angle. (Note that practically speaking, bend angles greater than $\frac{3\pi}{2}$ (270 degrees) can be problematic. Beyond this, the fringe fields from the entrance and exit pole faces could start to interfere, so be careful when setting up bend angles greater than this. An angle greater than or equal to 2π (360 degrees) is not allowed.)

K0 Field amplitude in y direction (Tesla). If the ANGLE attribute is set, K0 is ignored.

K0S Field amplitude in x direction (Tesla). If the ANGLE attribute is set, K0S is ignored.

GREATERTHANPI If setting the field strength using the K0 and/or K0S attributes and the intent is a bend angle greater than π , then this attribute must be set to TRUE. (This is a consequence of using the chord length for the magnet length.)

K1 Field gradient index of the magnet, $K_1 = -\frac{R}{B_y} \frac{\partial B_y}{\partial x}$, where R is the bend radius as defined in Figure 8.3.

E1 Entrance edge angle (radians). Figure 8.3 shows the definition of a positive entrance edge angle.

E2 Exit edge angle in radians. Figure 8.3 shows the definition of a positive exit edge angle.

ROTATION Rotation of magnet about its z axis (radians). Setting this angle allows us to achieve a bend in any direction in the x/y plane.

BETA Rotation of magnet about its x axis (radians). See Figure 8.2.

DESIGNENERGY Energy of the bend reference particle (eV). The reference particle travels the path shown in Figure 8.3.

FMAPFN Name of the field map for the magnet. Currently maps in the T7 format of type 1DProfile1 can be used (see C.11). The default option for this attribute is FMAPN = "1DPROFILE1-DEFAULT" (see 8.4.5). The field map is used to describe the fringe fields of the magnet (see ??).

8.4.3 RBend and SBend Examples (OPAL-T)

Describing an RBEND or an SBEND in an OPAL-T simulation requires effectively identical commands. There are only slight differences between the two. The `L` attribute has a different definition for the two types of bends (see 8.4.1 and 8.4.2), and an SBEND has additional attributes that have no effect on an RBEND (`E2` and `GREATERTHANPI`, see 8.4.2). Therefore, in this section, we will give several examples of how to implement a bend, using the RBEND and SBEND commands interchangeably. The understanding is that the command formats are essentially the same.

When implementing an RBEND or SBEND in an OPAL-T simulation, it is important to note the following:

1. Internally OPAL-T treats all bends as positive, as defined by Figures 8.1 and 8.3. Bends in other directions within the x/y plane are accomplished by rotating a positive bend about its z axis.
2. If the `ANGLE` attribute is set to a non-zero value, the `K0` and `K0S` attributes will be ignored.
3. When using the `ANGLE` attribute to define a bend, the actual beam will always be bent through that angle, regardless of the bend reference energy as set by the `DESIGNENERGY` attribute. (The bend field strength will be adjusted to ensure this happens.)
4. If the bend strength is set by `K0` and/or `K0S`, then the actual beam will be bent a different angle if its energy does not correspond to the `DESIGNENERGY` of the bend.
5. Internally the bend geometry is setup based on the ideal reference trajectory, as shown in Figures 8.1 and 8.3.
6. If the default field map, “1DPROFILE-DEFAULT” (see 8.4.5), is used, the fringe fields will be adjusted so that the effective length of the real, soft edge magnet matches the ideal, hard edge bend that is defined by the reference trajectory.
7. The length of the bend field can be set either using the `L` attribute or in the field map field file. We will go over these two different options below. However, it is important to note that if `L` is set to a non-zero value it will trump the value from the field file. (In turn, this can lead to strange and incorrect results.) Also note that *the `L` attribute must be set when using the default field map, “1DPROFILE-DEFAULT” (see 8.4.5).*

For the rest of this section, we will give several examples of how to input bends in an OPAL-T simulation. We will start with a simple example using the `ANGLE` attribute to set the bend strength and using the default field map (see 8.4.5) for describing the magnet fringe fields (see ??):

```
Bend: RBend, ANGLE = 30.0 * Pi / 180.0,
          FMAPFN = "1DPROFILE1-DEFAULT",
          ELEMEDGE = 0.25,
          DESIGNENERGY = 10.0E6,
          L = 0.5,
          GAP = 0.02;
```

This is a definition of a simple RBEND that bends the beam in a positive direction 30 degrees (towards the negative x axis as if Figure 8.1). It has a design energy of 10 MeV, a length of 0.5 m, a vertical gap of 2 cm and a 0 degree entrance edge angle. (Therefore the exit edge angle is 30 degrees.) We are using the default, internal field map “1DPROFILE1-DEFAULT” (see 8.4.5) which describes the magnet fringe fields (see ??). When OPAL is run, you will get the following output (assuming an electron beam) for this RBEND definition:

```
RBend > Reference Trajectory Properties
RBend > =====
RBend >
RBend > Bend angle magnitude:      0.523599 rad (30 degrees)
RBend > Entrance edge angle:       0 rad (0 degrees)
RBend > Exit edge angle:           0.523599 rad (30 degrees)
RBend > Bend design radius:        1 m
```

```

RBend > Bend design energy:      1e+07 eV
RBend >
RBend > Bend Field and Rotation Properties
RBend > =====
RBend >
RBend > Field amplitude:          -0.0350195 T
RBend > Field index (gradient):   0 m^-1
RBend > Rotation about x axis:    0 rad (0 degrees)
RBend > Rotation about y axis:    0 rad (0 degrees)
RBend > Rotation about z axis:    0 rad (0 degrees)
RBend >
RBend > Reference Trajectory Properties Through Bend Magnet with Fringe Fields
RBend > =====
RBend >
RBend > Reference particle is bent: 0.523599 rad (30 degrees) in x plane
RBend > Reference particle is bent: 0 rad (0 degrees) in y plane

```

The first section of this output gives the properties of the reference trajectory like that described in Figure 8.1. From the value of `ANGLE` and the length, `L`, of the magnet, OPAL calculates the 10 MeV reference particle trajectory radius, `R`. From the bend geometry and the entrance angle (0 in this case), the exit angle is calculated.

The second section gives the field amplitude of the bend and its gradient (quadrupole focusing component), given the particle charge ($-e$ in this case so the amplitude is negative to get a positive bend direction). Also listed is the rotation of the magnet about the various axes.

Of course, in the actual simulation the particles will not see a hard edge bend magnet, but rather a soft edge magnet with fringe fields described by the `RBEND` field map file `FMAPFN` (see ??). So, once the hard edge bend/reference trajectory is determined, OPAL then includes the fringe fields in the calculation. When the user chooses to use the default field map, OPAL will automatically adjust the position of the fringe fields appropriately so that the soft edge magnet is equivalent to the hard edge magnet described by the reference trajectory. To check that this was done properly, OPAL integrates the reference particle through the final magnet description with the fringe fields included. The result is shown in the final part of the output. In this case we see that the soft edge bend does indeed bend our reference particle through the correct angle.

What is important to note from this first example, is that it is this final part of the bend output that tells you the actual bend angle of the reference particle.

In this next example, we merely rewrite the first example, but use `K0` to set the field strength of the `RBEND`, rather than the `ANGLE` attribute:

```

Bend: RBend, K0 = -0.0350195,
      FMAPFN = "1DPROFILE1-DEFAULT",
      ELEMEDGE = 0.25,
      DESIGNENERGY = 10.0E6,
      L = 0.5,
      GAP = 0.02;

```

The output from OPAL now reads as follows:

```

RBend > Reference Trajectory Properties
RBend > =====
RBend >
RBend > Bend angle magnitude:      0.523599 rad (30 degrees)
RBend > Entrance edge angle:      0 rad (0 degrees)
RBend > Exit edge angle:          0.523599 rad (30 degrees)
RBend > Bend design radius:       0.999999 m
RBend > Bend design energy:       1e+07 eV
RBend >
RBend > Bend Field and Rotation Properties
RBend > =====
RBend >

```

```

RBend > Field amplitude:      -0.0350195 T
RBend > Field index (gradient): 0 m^-1
RBend > Rotation about x axis: 0 rad (0 degrees)
RBend > Rotation about y axis: 0 rad (0 degrees)
RBend > Rotation about z axis: 0 rad (0 degrees)
RBend >
RBend > Reference Trajectory Properties Through Bend Magnet with Fringe Fields
RBend > =====
RBend >
RBend > Reference particle is bent: 0.5236 rad (30.0001 degrees) in x plane
RBend > Reference particle is bent: 0 rad (0 degrees) in y plane

```

The output is effectively identical, to within a small numerical error.

Now, let us modify this first example so that we bend instead in the negative x direction. There are several ways to do this:

1.

```
Bend: RBend, ANGLE = -30.0 * Pi / 180.0,
    FMAPFN = "1DPROFILE1-DEFAULT",
    ELEMEDGE = 0.25,
    DESIGNENERGY = 10.0E6,
    L = 0.5,
    GAP = 0.02;
```

2.

```
Bend: RBend, ANGLE = 30.0 * Pi / 180.0,
    FMAPFN = "1DPROFILE1-DEFAULT",
    ELEMEDGE = 0.25,
    DESIGNENERGY = 10.0E6,
    L = 0.5,
    GAP = 0.02,
    ROTATION = Pi;
```

3.

```
Bend: RBend, K0 = 0.0350195,
    FMAPFN = "1DPROFILE1-DEFAULT",
    ELEMEDGE = 0.25,
    DESIGNENERGY = 10.0E6,
    L = 0.5,
    GAP = 0.02;
```

4.

```
Bend: RBend, K0 = -0.0350195,
    FMAPFN = "1DPROFILE1-DEFAULT",
    ELEMEDGE = 0.25,
    DESIGNENERGY = 10.0E6,
    L = 0.5,
    GAP = 0.02,
    ROTATION = Pi;
```

In each of these cases, we get the following output for the bend (to within small numerical errors).

```

RBend > Reference Trajectory Properties
RBend > =====
RBend >
RBend > Bend angle magnitude:    0.523599 rad (30 degrees)
RBend > Entrance edge angle:    0 rad (0 degrees)
RBend > Exit edge angle:        0.523599 rad (30 degrees)
RBend > Bend design radius:     1 m
RBend > Bend design energy:     1e+07 eV

```

```

RBend >
RBend > Bend Field and Rotation Properties
RBend > =====
RBend >
RBend > Field amplitude:          -0.0350195 T
RBend > Field index (gradient):  -0 m^-1
RBend > Rotation about x axis:    0 rad (0 degrees)
RBend > Rotation about y axis:    0 rad (0 degrees)
RBend > Rotation about z axis:    3.14159 rad (180 degrees)
RBend >
RBend > Reference Trajectory Properties Through Bend Magnet with Fringe Fields
RBend > =====
RBend >
RBend > Reference particle is bent: -0.523599 rad (-30 degrees) in x plane
RBend > Reference particle is bent: 0 rad (0 degrees) in y plane

```

In general, we suggest to always define a bend in the positive x direction (as in Figure 8.1) and then use the ROTATION attribute to bend in other directions in the x/y plane (as in examples 2 and 4 above).

As a final RBEND example, here is a suggested format for the four bend definitions if one were implementing a four dipole chicane:

```

Bend1: RBend, ANGLE = 20.0 * Pi / 180.0,
           E1 = 0.0,
           FMAPFN = "1DPROFILE1-DEFAULT",
           ELEMEDGE = 0.25,
           DESIGNENERGY = 10.0E6,
           L = 0.25,
           GAP = 0.02,
           ROTATION = Pi;

Bend2: RBend, ANGLE = 20.0 * Pi / 180.0,
           E1 = 20.0 * Pi / 180.0,
           FMAPFN = "1DPROFILE1-DEFAULT",
           ELEMEDGE = 1.0,
           DESIGNENERGY = 10.0E6,
           L = 0.25,
           GAP = 0.02,
           ROTATION = 0.0;

Bend3: RBend, ANGLE = 20.0 * Pi / 180.0,
           E1 = 0.0,
           FMAPFN = "1DPROFILE1-DEFAULT",
           ELEMEDGE = 1.5,
           DESIGNENERGY = 10.0E6,
           L = 0.25,
           GAP = 0.02,
           ROTATION = 0.0;

Bend4: RBend, ANGLE = 20.0 * Pi / 180.0,
           E1 = 20.0 * Pi / 180.0,
           FMAPFN = "1DPROFILE1-DEFAULT",
           ELEMEDGE = 2.25,
           DESIGNENERGY = 10.0E6,
           L = 0.25,
           GAP = 0.02,
           ROTATION = Pi;

```

Up to now, we have only given examples of RBEND definitions. If we replaced “RBend” in the above examples with “SBend”, we would still be defining valid OPAL-T bends. In fact, by adjusting the L attribute according to Sections 8.4.1 and 8.4.2, and by adding the appropriate definitions of the E2 attribute, we could even get identical results using SBENDS instead of RBENDS. (As we said, the two bends are very similar in command format.)

Up till now, we have only used the default field map. Custom field maps can also be used. There are two different options in this case (see Section C.11):

1. Field map defines fringe fields and magnet length.
2. Field map defines fringe fields only.

The first case describes how field maps were used in previous versions of OPAL (and can still be used in the current version). The second option is new to OPAL 1.2.0 and it has a couple of advantages:

1. Because only the fringe fields are described, the length of the magnet must be set using the `L` attribute. In turn, this means that the same field map can be used by many bend magnets with different lengths (assuming they have equivalent fringe fields). By contrast, if the magnet length is set by the field map, one must generate a new field map for each dipole of different length even if the fringe fields are the same.
2. We can adjust the position of the fringe field origin relative to the entrance and exit points of the magnet (see ?? and C.11.) This gives us another degree of freedom for describing the fringe fields, allowing us to adjust the effective length of the magnet.

We will now give examples of how to use a custom field map, starting with the first case where the field map describes the fringe fields and the magnet length. Assume we have the following `1DProfile1` field map:

```
1DProfile1 1 1 2.0
-10.0  0.0  10.0 1
 15.0  25.0 35.0 1
 0.00000E+00
 2.00000E+00
 0.00000E+00
 2.00000E+00
```

We can use this field map to define the following bend (note we are now using the `SBEND` command):

```
Bend: SBend, ANGLE = 60.0 * Pi / 180.0,
      E1 = -10.0 * Pi / 180.0,
      E2 = 20.0 * Pi / 180.0,
      FMAPFN = "TEST-MAP.T7",
      ELEMEDGE = 0.25,
      DESIGNENERGY = 10.0E6,
      GAP = 0.02;
```

Notice that we do not set the magnet length using the `L` attribute. (In fact, we don't even include it. If we did and set it to a non-zero value, the exit fringe fields of the magnet would not be correct.) This input gives the following output:

```
SBend > Reference Trajectory Properties
SBend > =====
SBend >
SBend > Bend angle magnitude:      1.0472 rad (60 degrees)
SBend > Entrance edge angle:      -0.174533 rad (-10 degrees)
SBend > Exit edge angle:          0.349066 rad (20 degrees)
SBend > Bend design radius:        0.25 m
SBend > Bend design energy:        1e+07 eV
SBend >
SBend > Bend Field and Rotation Properties
SBend > =====
SBend >
SBend > Field amplitude:            -0.140385 T
```

```

SBend > Field index (gradient): 0 m^-1
SBend > Rotation about x axis: 0 rad (0 degrees)
SBend > Rotation about y axis: 0 rad (0 degrees)
SBend > Rotation about z axis: 0 rad (0 degrees)
SBend >
SBend > Reference Trajectory Properties Through Bend Magnet with Fringe Fields
SBend > =====
SBend >
SBend > Reference particle is bent: 1.0472 rad (60 degrees) in x plane
SBend > Reference particle is bent: 0 rad (0 degrees) in y plane

```

Because we set the bend strength using the `ANGLE` attribute, the magnet field strength is automatically adjusted so that the reference particle is bent exactly `ANGLE` radians when the fringe fields are included. (Lower output.)

Now we will illustrate the case where the magnet length is set by the `L` attribute and only the fringe fields are described by the field map. We change the `TEST-MAP.T7` file to:

```

IDProfile1 1 1 2.0
-10.0 0.0 10.0 1
-10.0 0.0 10.0 1
0.00000E+00
2.00000E+00
0.00000E+00
2.00000E+00

```

and change the bend input to:

```

Bend: SBend, ANGLE = 60.0 * Pi / 180.0,
      E1 = -10.0 * Pi / 180.0,
      E2 = 20.0 * Pi / 180.0,
      FMAPFN = "TEST-MAP.T7",
      ELEMEDGE = 0.25,
      DESIGNENERGY = 10.0E6,
      L = 0.25,
      GAP = 0.02;

```

This results in the same output as the previous example, as we expect.

```

SBend > Reference Trajectory Properties
SBend > =====
SBend >
SBend > Bend angle magnitude: 1.0472 rad (60 degrees)
SBend > Entrance edge angle: -0.174533 rad (-10 degrees)
SBend > Exit edge angle: 0.349066 rad (20 degrees)
SBend > Bend design radius: 0.25 m
SBend > Bend design energy: 1e+07 eV
SBend >
SBend > Bend Field and Rotation Properties
SBend > =====
SBend >
SBend > Field amplitude: -0.140385 T
SBend > Field index (gradient): 0 m^-1
SBend > Rotation about x axis: 0 rad (0 degrees)
SBend > Rotation about y axis: 0 rad (0 degrees)
SBend > Rotation about z axis: 0 rad (0 degrees)
SBend >
SBend > Reference Trajectory Properties Through Bend Magnet with Fringe Fields
SBend > =====
SBend >
SBend > Reference particle is bent: 1.0472 rad (60 degrees) in x plane
SBend > Reference particle is bent: 0 rad (0 degrees) in y plane

```

As a final example, let us now use the previous field map with the following input:

```
Bend: SBend, K0 = -0.1400778,
      E1 = -10.0 * Pi / 180.0,
      E2 = 20.0  Pi / 180.0,
      FMAPFN = "TEST-MAP.T7",
      ELEMEDGE = 0.25,
      DESIGNENERGY = 10.0E6,
      L = 0.25,
      GAP = 0.02;
```

Instead of setting the bend strength using ANGLE, we use K0. This results in the following output:

```
SBend > Reference Trajectory Properties
SBend > =====
SBend >
SBend > Bend angle magnitude:      1.0472 rad (60 degrees)
SBend > Entrance edge angle:      -0.174533 rad (-10 degrees)
SBend > Exit edge angle:          0.349066 rad (20 degrees)
SBend > Bend design radius:        0.25 m
SBend > Bend design energy:        1e+07 eV
SBend >
SBend > Bend Field and Rotation Properties
SBend > =====
SBend >
SBend > Field amplitude:            -0.140078 T
SBend > Field index (gradient):    0 m^-1
SBend > Rotation about x axis:      0 rad (0 degrees)
SBend > Rotation about y axis:      0 rad (0 degrees)
SBend > Rotation about z axis:      0 rad (0 degrees)
SBend >
SBend > Reference Trajectory Properties Through Bend Magnet with Fringe Fields
SBend > =====
SBend >
SBend > Reference particle is bent: 1.04491 rad (59.8688 degrees) in x plane
SBend > Reference particle is bent: 0 rad (0 degrees) in y plane
```

In this case, the bend angle for the reference trajectory in the first section of the output no longer matches the reference trajectory bend angle from the lower section (although the difference is small). The reason is that the path of the reference particle through the real magnet (with fringe fields) no longer matches the ideal trajectory. (The effective length of the real magnet is not quite the same as the hard edged magnet for the reference trajectory.)

We can compensate for this by changing the field map file TEST-MAP.T7 file to:

```
1DProfile1 1 1 2.0
-10.0 -0.03026 10.0 1
-10.0 0.03026 10.0 1
0.00000E+00
2.00000E+00
0.00000E+00
2.00000E+00
```

We have moved the Enge function origins (Section ??) outward from the entrance and exit faces of the magnet (Section C.11) by 0.3026 mm. This has the effect of making the effective length of the soft edge magnet longer. When we do this, the same input:

```
Bend: SBend, K0 = -0.1400778,
      E1 = -10.0 * Pi / 180.0,
      E2 = 20.0  Pi / 180.0,
```

```

FMAPFN = "TEST-MAP.T7",
ELEMEDGE = 0.25,
DESIGNENERGY = 10.0E6,
L = 0.25,
GAP = 0.02;

```

produces

```

SBend > Reference Trajectory Properties
SBend > =====
SBend >
SBend > Bend angle magnitude:      1.0472 rad (60 degrees)
SBend > Entrance edge angle:      -0.174533 rad (-10 degrees)
SBend > Exit edge angle:          0.349066 rad (20 degrees)
SBend > Bend design radius:       0.25 m
SBend > Bend design energy:       1e+07 eV
SBend >
SBend > Bend Field and Rotation Properties
SBend > =====
SBend >
SBend > Field amplitude:           -0.140078 T
SBend > Field index (gradient):    0 m^-1
SBend > Rotation about x axis:     0 rad (0 degrees)
SBend > Rotation about y axis:     0 rad (0 degrees)
SBend > Rotation about z axis:     0 rad (0 degrees)
SBend >
SBend > Reference Trajectory Properties Through Bend Magnet with Fringe Fields
SBend > =====
SBend >
SBend > Reference particle is bent: 1.0472 rad (60 degrees) in x plane
SBend > Reference particle is bent: 0 rad (0 degrees) in y plane

```

Now we see that the bend angle for the ideal, hard edge magnet, matches the bend angle of the reference particle through the soft edge magnet. In other words, the effective length of the soft edge, real magnet is the same as the hard edge magnet described by the reference trajectory.

8.4.4 Bend Fields from 1D Field Maps (OPAL-T)

So far we have described how to setup an RBEND or SBEND element, but have not explained how OPAL-T uses this information to calculate the magnetic field. The field of both types of magnets is divided into three regions:

1. Entrance fringe field.
2. Central field.
3. Exit fringe field.

This can be seen clearly in Figure C.7.

The purpose of the 1DProfile1 field map (see C.11) associated with the element is to define the Enge functions (Equation 8.1) that model the entrance and exit fringe fields. To model a particular bend magnet, one must fit the field profile along the midplane of the magnet perpendicular to its face for the entrance and exit fringe fields to the Enge function:

$$F(z) = \frac{1}{1 + e^{\sum_{n=0}^{N_{order}} c_n (z/D)^n}} \quad (8.1)$$

where D is the full gap of the magnet, N_{order} is the Enge function order and z is the distance from the origin of the Enge function perpendicular to the edge of the dipole. The origin of the Enge function, the order of the Enge

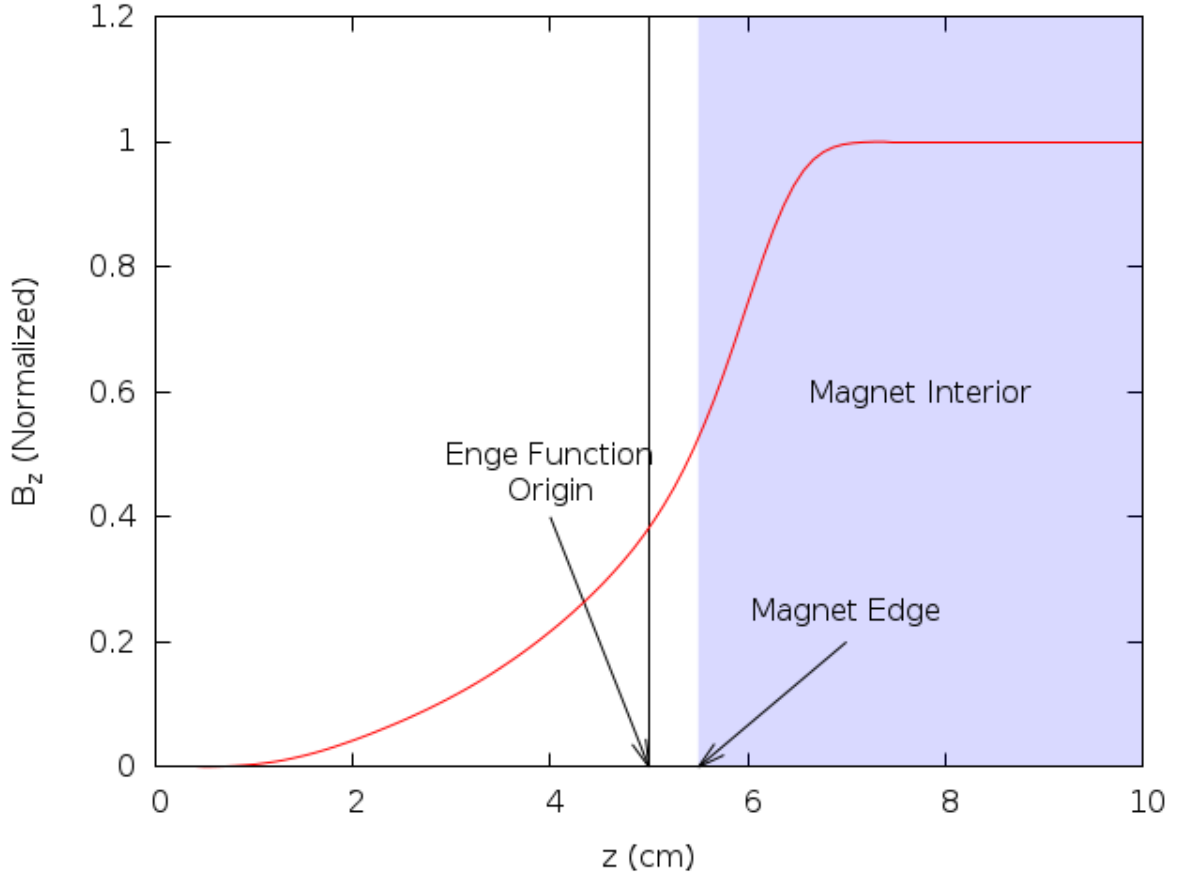


Figure 8.4: Plot of the entrance fringe field of a dipole magnet along the mid-plane, perpendicular to its entrance face. The field is normalized to 1.0. In this case, the fringe field is described by an Enge function (see Equation 8.1) with the parameters from the default `1DProfile1` field map described in 8.4.5. The exit fringe field of this magnet is the mirror image.

function, N_{order} , and the constants c_0 to $c_{N_{order}}$ are free parameters that are chosen so that the function closely approximates the fringe region of the magnet being modeled. An example of the entrance fringe field is shown in Figure 8.4.

Let us assume we have a correctly defined positive RBEND or SBEND element as illustrated in Figures 8.1 and 8.3. (As already stated, any bend can be described by a rotated positive bend.) OPAL-T then has the following

information:

B_0 = Field amplitude (T)

R = Bend radius (m)

$n = -\frac{R}{B_y} \frac{\partial B_y}{\partial x}$ (Field index, set using the parameter K1)

$$F(z) = \begin{cases} F_{entrance}(z_{entrance}) \\ F_{center}(z_{center}) = 1 \\ F_{exit}(z_{exit}) \end{cases}$$

Here, we have defined an overall Enge function, $F(z)$, with three parts: entrance, center and exit. The exit and entrance fringe field regions have the form of Equation 8.1 with parameters defined by the `1DProfile1` field map file given by the element parameter `FMAPFN`. Defining the coordinates:

$y \equiv$ Vertical distance from magnet mid-plane

$\Delta_x \equiv$ Perpendicular distance to reference trajectory (see Figures 8.1 and 8.3)

$$\Delta_z \equiv \begin{cases} \text{Distance from entrance Enge function origin perpendicular to magnet entrance face.} \\ \text{Not defined, Enge function is always 1 in this region.} \\ \text{Distance from exit Enge function origin perpendicular to magnet exit face.} \end{cases}$$

using the conditions

$$\nabla \cdot \vec{B} = 0$$

$$\nabla \times \vec{B} = 0$$

and making the definitions:

$$F'(z) \equiv \frac{dF(z)}{dz}$$

$$F''(z) \equiv \frac{d^2F(z)}{dz^2}$$

$$F'''(z) \equiv \frac{d^3F(z)}{dz^3}$$

we can expand the field off axis, with the result:

$$\begin{aligned}
B_x(\Delta_x, y, \Delta_z) &= -\frac{B_0 \frac{n}{R}}{\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}}} e^{-\frac{n}{R}\Delta_x} \sin \left[\left(\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}} y \right) F(\Delta_z) \right] \\
B_y(\Delta_x, y, \Delta_z) &= B_0 e^{-\frac{n}{R}\Delta_x} \cos \left[\left(\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}} y \right) F(\Delta_z) \right] \\
B_z(\Delta_x, y, \Delta_z) &= B_0 e^{-\frac{n}{R}\Delta_x} \left\{ \frac{F'(\Delta_z)}{\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}}} \sin \left[\left(\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}} y \right) \right] \right. \\
&\quad - \frac{1}{2\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}}} \left(F'''(\Delta_z) - \frac{F'(\Delta_z)F''(\Delta_z)}{F(\Delta_z)} \right) \left[\frac{\sin \left[\left(\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}} y \right) \right]}{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}} \right. \\
&\quad \left. \left. - y \frac{\cos \left[\left(\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}} y \right) \right]}{\sqrt{\frac{n^2}{R^2} + \frac{F''(\Delta_z)}{F(\Delta_z)}}} \right] \right\}
\end{aligned}$$

These expression are not well suited for numerical calculation, so, we expand them about y to $O(y^2)$ to obtain:

- In fringe field regions:

$$\begin{aligned}
B_x(\Delta_x, y, \Delta_z) &\approx -B_0 \frac{n}{R} e^{-\frac{n}{R}\Delta_x} y \\
B_y(\Delta_x, y, \Delta_z) &\approx B_0 e^{-\frac{n}{R}\Delta_x} \left[F(\Delta_z) - \left(\frac{n^2}{R^2} F(\Delta_z) + F''(\Delta_z) \right) \frac{y^2}{2} \right] \\
B_z(\Delta_x, y, \Delta_z) &\approx B_0 e^{-\frac{n}{R}\Delta_x} y F'(\Delta_z)
\end{aligned}$$

- In central region:

$$\begin{aligned}
B_x(\Delta_x, y, \Delta_z) &\approx -B_0 \frac{n}{R} e^{-\frac{n}{R}\Delta_x} y \\
B_y(\Delta_x, y, \Delta_z) &\approx B_0 e^{-\frac{n}{R}\Delta_x} \left[1 - \frac{n^2}{R^2} \frac{y^2}{2} \right] \\
B_z(\Delta_x, y, \Delta_z) &\approx 0
\end{aligned}$$

These are the expressions OPAL-T uses to calculate the field inside an RBEND or SBEND. First, a particle's position inside the bend is determined (entrance region, center region, or exit region). Depending on the region, OPAL-T then determines the values of Δ_x , y and Δ_z , and then calculates the field values using the above expressions.

8.4.5 Default Field Map (OPAL-T)

Rather than force users to calculate the field of a dipole and then fit that field to find Enge coefficients for the dipoles in their simulation, we have a default set of values we use from [44] that are set when the default field map, "1DPROFILE1-DEFAULT" is used:

$$\begin{aligned}
c_0 &= 0.478959 \\
c_1 &= 1.911289 \\
c_2 &= -1.185953 \\
c_3 &= 1.630554 \\
c_4 &= -1.082657 \\
c_5 &= 0.318111
\end{aligned}$$

The same values are used for both the entrance and exit regions of the magnet. In general they will give good results. (Of course, at some point as a beam line design becomes more advanced, one will want to find Enge coefficients that fit the actual magnets that will be used in a given design.)

The default field map is the equivalent of the following custom `1DProfile1` (see C.11 for an explanation of the field map format) map:

```

1DProfile1 5 5 2.0
-10.0 0.0 10.0 1
-10.0 0.0 10.0 1
 0.478959
 1.911289
-1.185953
 1.630554
-1.082657
 0.318111
 0.478959
 1.911289
-1.185953
 1.630554
-1.082657
 0.318111

```

As one can see, the default magnet gap for “`1DPROFILE1-DEFAULT`” is set to 2.0 cm. This value can be overridden by the `GAP` attribute of the magnet (see 8.4.1 and 8.4.2).

8.4.6 SBend3D (OPAL-CYCL)

The SBend3D element enables definition of a bend from 3D field maps. This can be used in conjunction with the RINGDEFINITION element to make a ring for tracking through OPAL-CYCL.

```
label: SBEND3D, FMAPFN=string, LENGTH_UNITS=real, FIELD_UNITS=real;
```

FMAPFN The field map filename.

LENGTH_UNITS Units for length (set to 1.0 for units in mm, 10.0 for units in cm, etc).

FIELD_UNITS Units for field (set to 1.0 for units in T, 0.001 for units in mT, etc).

Field maps are defined using Cartesian coordinates but in a polar geometry. Field point positions and field values are written on a grid in (r, y, ϕ) . The first 8 lines are presumed to be header material and are ignored.

Below is an example of a SBEND3D which loads a field map file named `fdf-tosca-field-map.table`. Positions have units of cm and fields units of Gauss.

```
triplet: SBEND3D, FMAPFN="fdf-tosca-field-map.table", LENGTH_UNITS=10., FIELD_UNITS=-1e-4;
```

A sample field map file is as follows:


```

      422280      422280      422280      1
1 X [LENGU]
2 Y [LENGU]
3 Z [LENGU]
4 BX [FLUXU]
5 BY [FLUXU]
6 BZ [FLUXU]
0
194.01470 0.0000000 80.363520 0.68275932346E-07 -5.3752492577 0.28280706805E-07
194.36351 0.0000000 79.516210 0.42525693524E-07 -5.3827955117 0.17681348191E-07
194.70861 0.0000000 78.667380 0.19766168358E-07 -5.4350026348 0.82540823165E-08
<continues>
```

This is a restricted feature: OPAL-CYCL.

8.5 Quadrupole

```
label:QUADRUPOLE, TYPE=string, APERTURE=real-vector,
      L=real, K1=real, K1S=real;
```

The reference system for a quadrupole is a Cartesian coordinate system. This is a restricted feature:  OPAL-CYCL. A QUADRUPOLE has three real attributes:

L The quadrupole length (default: 0 m).

K1 The normal quadrupole component $K_1 = \frac{\partial B_y}{\partial x}$.

The default is 0 T m⁻². The component is positive, if B_y is positive on the positive x -axis.

K1S The skew quadrupole component. $K_{1s} = \frac{\partial B_x}{\partial x}$.

The default is 0 T m⁻². The component is negative, if B_x is positive on the positive x -axis.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

Example:

```
QP1: Quadrupole, L=1.20, ELEMEDGE=-0.5265,
      FMAPFN="1T1.T7", K1=0.11;
```

8.6 Sextupole

```
label: SEXTUPOLE, TYPE=string, APERTURE=real-vector,
      L=real, K2=real, K2S=real;
```

A SEXTUPOLE has three real attributes:

L The sextupole length (default: 0 m).

K2 The normal sextupole component $K_2 = \frac{\partial^2 B_y}{\partial x^2}$. The default is 0 T m⁻³. The component is positive, if B_y is positive on the x -axis.

K2S The skew sextupole component $K_{2s} = \frac{\partial^2 B_x}{\partial x^2}$. The default is 0 T m⁻³. The component is negative, if B_x is positive on the x -axis.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

Example:

```
S:SEXTUPOLE, L=0.4, K2=0.00134;
```

The reference system for a sextupole is a Cartesian coordinate system.

8.7 Octupole

```
label:OCTUPOLE, TYPE=string, APERTURE=real-vector,
      L=real, K3=real, K3S=real;
```

An OCTUPOLE has three real attributes:

L The octupole length (default: 0 m).

K3 The normal sextupole component $K_3 = \frac{\partial^3 B_y}{\partial x^3}$. The default is 0 T m⁻⁴. The component is positive, if B_y is positive on the positive x -axis.

K3S The skew sextupole component $K_{3s} = \frac{\partial^3 B_x}{\partial x^3}$. The default is 0 T m⁻⁴. The component is negative, if B_x is positive on the positive x -axis.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

Example:

```
O3:OCTUPOLE, L=0.3, K3=0.543;
```

The reference system for an octupole is a Cartesian coordinate system.

8.8 General Multipole

A MULTIPOLE is in OPAL-T is of arbitrary order.

```
label:MULTIPOLE, TYPE=string, APERTURE=real-vector,
      L=real, KN=real-vector, KS=real-vector;
```

L The multipole length (default: 0 m).

KN A real vector (see §6.13), containing the normal multipole coefficients, $K_n = \frac{\partial^n B_y}{\partial x^n}$ (default is 0 T m⁻ⁿ). A component is positive, if B_y is positive on the positive x -axis.

KS A real vector (see §6.13), containing the skew multipole coefficients, $K_{3n} = \frac{\partial^3 B_n}{\partial x^n}$ (default is 0 T m⁻ⁿ). A component is negative, if B_x is positive on the positive x -axis.

The multipole coefficients are defined in the same units than the coefficients of the QUADRUPOLE and SEXTUPOLE, etc. The order n is unlimited, but all components up to the maximum must be given, even if they are zero. The number of poles of each component is $(2n + 2)$. Superposition of many multipole components is permitted. The reference system for a multipole is a Cartesian coordinate system.

The following example is equivalent to the quadruple example in 8.5.

```
M27:MULTIPOLE, L=1, ELEMEDGE=3.8, KN={0.0,0.11}, KS={0.0,0.0, 0.42};
```

A multipole has no effect on the reference orbit, i.e. the reference system at its exit is the same as at its entrance. Use the dipole component only to model a defective multipole.

8.9 Solenoid

```
label:SOLENOID, TYPE=string, APERTURE=real-vector,
      L=real, KS=real;
```

A SOLENOID has two real attributes:

L The length of the solenoid (default: 0 m)

KS The solenoid strength K_s (default: 0 rad/m). For positive KS and positive particle charge, the solenoid field points in the direction of increasing s .

The reference system for a solenoid is a Cartesian coordinate system. Using a solenoid in OPAL-t mode, the following additional parameters are defined:

FMAPFN Field maps must be specified.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

Example:

```
SP1: Solenoid, L=1.20, ELEMEDGE=-0.5265, KS=0.11,
      FMAPFN="1T1.T7";
```

8.10 Cyclotron

```
label:CYCLOTRON, TYPE=string, CYHARMON=int,
      PHIINIT=real, PRINIT=real, RINIT=real,
      SYMMETRY=real, RFFREQ=real, FMAPFN=string;
```

A CYCLOTRON object includes the main characteristics of a cyclotron, the magnetic field, and also the initial condition of the injected reference particle, and it has currently the following attributes:

TYPE The data format of field map, Currently three formats are implemented: CARBONCYCL, CYCIAE, AVFEQ, FFAG, BANDRF and default PSI format. For the details of their data format, please read Section 5.4.

CYHARMON The hamonic number of the cyclotron h .

RFFREQ The RF system f_{rf} (unit:MHz, default: 0). The particle revolution frequency $f_{rev} = f_{rf} / h$.

FMAPFN Filename for the magnetic field map.

SYMMETRY Defines symmetrical fold number of the B field map data.

RINIT The initial radius of the reference particle (unit: mm, default: 0)

PHIINIT The initial azimuth of the reference particle (unit: degree, default: 0)

ZINIT The initial axial position of the reference particle (unit: mm, default: 0)

PRINIT Initial radial momentum of the reference particle $P_r = \beta_r \gamma$ (default : 0)

PZINIT Initial axial momentum of the reference particle $P_z = \beta_z \gamma$ (default : 0)

MINZ The minimal vertical extent of the machine (unit: mm, default : -10000.0)

MAXZ The maximal vertical extent of the machine (unit: mm, default : 10000.0)

MINR Minimal radial extent of the machine (unit: mm, default : 0.0)

MAXR Minimal radial extent of the machine (unit: mm, default : 10000.0)

During the tracking, the particle (r, z, θ) will be deleted if $\text{MINZ} < z < \text{MAXZ}$ or $\text{MINR} < r < \text{MAXR}$, and it will be recorded in the ASCII file *inputfilename.loss*. Example:

```
ring: Cyclotron, TYPE="RING", CYHARMON=6, PHIINIT=0.0,
      PRINIT=-0.000240, RINIT=2131.4 , SYMMETRY=8.0,
      RFFREQ=50.650, FMAPFN="s03av.nar",
      MAXZ=10, MINZ=-10, MINR=0, MAXR=2500;
```

If TYPE is set to BANDRF, the 3D electric field map of RF cavity will be read from external h5part file and 4 extra arguments need to specified:

RFFMAPFN The filename for the electric fieldmap in h5part binary format.

RFPHI The Initial phase of the electric field map (rad)

ESCALE The maximal value of the electric fieldmap (MV/m)

SUPERPOSE An option whether all of the electric field maps are superposed, The is valid when more than one electric field map is read. (default: true)

Example for single electric field map:

```
COMET: Cyclotron, TYPE="BANDRF", CYHARMON=2, PHIINIT= -71.0,
PRINIT=pr0, RINIT= r0 , SYMMETRY=1.0, FMAPFN="Tosca_map.txt",
RFPHI=Pi, RFFREQ=72.0, RFPFN="efield.h5part",
ESCALE=1.06E-6;
```

We can have more than one RF field maps.

Example for multiple RF field maps:

```
COMET: Cyclotron, TYPE="BANDRF", CYHARMON=2, PHIINIT=-71.0,
PRINIT=pr0, RINIT=r0 , SYMMETRY=1.0, FMAPFN="Tosca_map.txt",
RFPHI= {Pi,0,Pi,0}, RFFREQ={72.0,72.0,72.0,72.0},
RFPFN={"e1.h5part", "e2.h5part", "e3.h5part", "e4.h5part"},
ESCALE={1.06E-6, 3.96E-6, 1.3E-6, 1.E-6}, SUPERPOSE=true;
```

In this example SUPERPOSE is set to true. Therefore, if a particle locates in multiple field regions, all the field maps are superposed. if SUPERPOSE is set to false, then only one field map, which has highest priority, is used to do interpolation for the particle tracking. The priority ranking is decided by their sequence in the list of RFPFN argument, i.e., "e1.h5part" has the highest priority and "e4.h5part" has the lowest priority.

Another method to model an RF cavity is to read the RF voltage profile in the RFCAVITY element (see Section 8.12) and make a momentum kick when a particle crosses the RF gap. In the center region of the compact cyclotron, the electric field shape is complicated and may make a significant impact on transverse beam dynamics. Hence a simple momentum kick is not enough and we need to read 3D field map to do precise simulation.

In additions, the simplified trim-coil field model is also implemented so as to do fine tuning on the magnetic field. A trim-coil can be defined by 4 arguments:

TCR1 The inner radius of the trim coil (mm)

TCR2 The outer radius of the trim coil (mm)

MBTC The maximal B field of trim coil (kG)

SLPTC The slope of the rising edge (kG/mm)

This is a restricted feature: OPAL-CYCL.

8.11 RingDefinition

```
label: RINGDEFINITION,
      RFFREQ=real, HARMONIC_NUMBER=real, IS_CLOSED=string, SYMMETRY=int,
      LAT_RINIT=real, LAT_PHIINIT=real, LAT_THETAINIT=real,
      BEAM_PHIINIT=real, BEAM_PRINIT=real, BEAM_RINIT=real;
```

A `RingDefinition` object contains the main characteristics of a generalised ring. The `RingDefinition` lists characteristics of the entire ring such as harmonic number together with the position of the initial element and the position of the reference trajectory.

The `RingDefinition` can be used in combination with `SBEND3D`, offsets and `VARIABLE_RF_CAVITY` elements to make up a complete ring.

RFFREQ Nominal RF frequency of the ring [MHz].

HARMONIC_NUMBER The harmonic number of the ring - i.e. number of bunches in a single pass.

SYMMETRY Azimuthal symmetry of the ring. Ring elements will be placed repeatedly `SYMMETRY` times.

IS_CLOSED Set to `FALSE` to disable checking for ring closure.

LAT_RINIT Radius of the first element placement in the lattice [mm].

LAT_PHIINIT Azimuthal angle of the first element placed in the lattice [degree].

LAT_THETAINIT Angle in the midplane relative to the ring tangent for placement of the first element [degree].

BEAM_RINIT Initial radius of the reference trajectory [mm].

BEAM_PHIINIT Initial azimuthal angle of the reference trajectory [degree].

BEAM_PRINIT Transverse momentum $\beta\gamma$ for the reference trajectory.

In the following example, we define a ring with radius 2.35 m and 4 cells.

```
ringdef: RINGDEFINITION, HARMONIC_NUMBER=6, LAT_RINIT=2350.0, LAT_PHIINIT=0.0,
      LAT_THETAINIT=0.0, BEAM_PHIINIT=0.0, BEAM_PRINIT=0.0,
      BEAM_RINIT=2266.0, SYMMETRY=4.0, RFFREQ=0.2;
```

8.11.1 Local Cartesian Offset

The `LOCAL_CARTESIAN_OFFSET` enables the user to place an object at an arbitrary position in the coordinate system of the preceding element. This enables drift spaces and placement of overlapping elements.

END_POSITION_X x position of the next element start in the coordinate system of the preceding element [mm].

END_POSITION_Y y position of the next element start in the coordinate system of the preceding element [mm].

END_NORMAL_X x component of the normal vector defining the placement of the next element in the coordinate system of the preceding element.

END_NORMAL_Y y component of the normal vector defining the placement of the next element in the coordinate system of the preceding element.

8.12 RF Cavities (OPAL-T and OPAL-CYCL)

For an RFCAVITY the three modes have four real attributes in common:

```
label:RFCAVITY, APERTURE=real-vector, L=real,
      VOLT=real, LAG=real;
```

L The length of the cavity (default: 0 m)

VOLT The peak RF voltage (default: 0 MV). The effect of the cavity is $\delta E = \text{VOLT} \cdot \sin(2\pi(\text{LAG} - \text{HARMON} \cdot f_0 t))$.

LAG The phase lag [rad] (default: 0).

8.12.1 OPAL-T mode

Using a RF Cavity in OPAL-T mode, the following additional parameters are defined:

FMAPFN Field maps in the T7 format can be specified.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

TYPE Type specifies STANDING [default] or SINGLE GAP structures.

FREQ Defines the frequency of the RF Cavity in units of MHz. A warning is issued when the frequency of the cavity card does not correspond to the frequency defined in the FMAPFN file. The frequency of the cavity card overrides the frequency defined in the FMAPFN file.

APVETO If TRUE this cavity will not be auto-phased, instead the actual phase on the element is used.

Example standing wave cavity which mimics a DC gun:

```
gun: RFCavity, L=0.018, VOLT=-131/(1.052*2.658),
      FMAPFN="1T3.T7", ELEMEDGE=0.00,
      TYPE="STANDING", FREQ=1.0e-6;
```

Example of a two frequency standing wave cavity:

```
rf1: RFCavity, L=0.54, VOLT=19.961, LAG=193.0/360.0,
      FMAPFN="1T3.T7", ELEMEDGE=0.129, TYPE="STANDING",
      FREQ=1498.956;
rf2: RFCavity, L=0.54, VOLT=6.250, LAG=136.0/360.0,
      FMAPFN="1T4.T7", ELEMEDGE=0.129, TYPE="STANDING",
      FREQ=4497.536;
```

8.12.2 OPAL-CYCL mode

Using a RF Cavity (standing wave) in OPAL-CYCL mode, the following parameters are defined:

FMAPFN Defines name of file which stores normalized voltage amplitude curve of cavity gap in ASCII format.
(See data format in Section 5.4)

VOLT Sets peak value of voltage amplitude curve in MV.

TYPE Defines Cavity type, SINGLEGAP represents cyclotron type cavity.

FREQ Sets the frequency of the RF Cavity in units of MHz.

RMIN Sets the radius of the cavity inner edge in mm.

RMAX Sets the radius of the cavity outer edge in mm.

ANGLE Sets the azimuthal position of the cavity in global frame in degree.

PDIS Set shift distance of cavity gap from center of cyclotron in mm. If its value is positive, the shift direction is clockwise, namely, shift towards the smaller azimuthal angle.

GAPWIDTH Set gap width of cavity in mm.

PHI0 Set initial phase of cavity in degree.

Example of a RF cavity of cyclotron:

```
rf0: RfCavity, VOLT=0.25796, FMAPFN="Cav1.dat",
      TYPE="SINGLE GAP", FREQ=50.637, RMIN = 350.0,
      RMAX = 3350.0, ANGLE=35.0, PDIS = 0.0,
      GAPWIDTH = 0.0, PHI0=phi01;
```

Fig. 8.5 shows the simplified geometry of a cavity gap and its parameters.

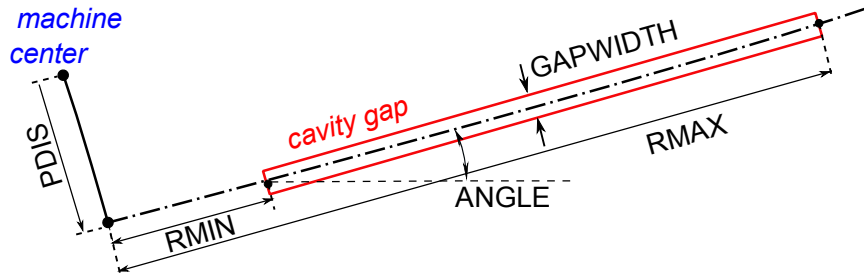


Figure 8.5: Schematic of the simplified geometry of a cavity gap and parameters

8.13 RF Cavities with Time Dependent Parameters

The `VARIABLE_RF_CAVITY` element can be used to define RF Cavities with Time Dependent Parameters in OPAL-CYCL mode. Variable RF Cavities must be placed using the `RingDefinition` element.

FREQUENCY_MODEL String naming the time dependence model of the cavity frequency, f [GHz].

AMPLITUDE_MODEL String naming the time dependence model of the cavity amplitude, E_0 [MV/m].

PHASE_MODEL String naming the time dependence model of the cavity phase offset, ϕ .

WIDTH Full width of the cavity [mm].

HEIGHT Full height of the cavity [mm].

L Full length of the cavity [mm].

The field inside the cavity is given by

$$\mathbf{E} = (0, 0, \mathbf{E}_0(t) \sin[2\pi f(t)t + \phi(t)]) \quad (8.2)$$

with no field outside the cavity boundary. There is no magnetic field or transverse dependence on electric field.

8.13.1 Time Dependence

The `POLYNOMIAL_TIME_DEPENDENCE` element is used to define time dependent parameters in RF cavities in terms of a 4th order polynomial.

P0 Constant term in the polynomial expansion.

P1 First order term in the polynomial expansion [ns^{-1}].

P2 Second order term in the polynomial expansion [ns^{-2}].

P3 Third order term in the polynomial expansion [ns^{-3}].

P4 Fourth order term in the polynomial expansion [ns^{-4}].

The polynomial is evaluated as

$$g(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3 + p_4 t^4. \quad (8.3)$$

8.13.2 Example

An example of a Variable Frequency RF cavity of cyclotron with polynomial time dependence of parameters is given below:

```

phi=2.*PI*0.25;

rf_p0=0.00158279;
rf_p1=9.02542e-10;
rf_p2=-1.96663e-16;
rf_p3=2.45909e-23;

RF_FREQUENCY: POLYNOMIAL_TIME_DEPENDENCE, P0=rf_p0, P1=rf_p1, P2=rf_p2, P3=rf_p3;
RF_AMPLITUDE: POLYNOMIAL_TIME_DEPENDENCE, P0=1.0;
RF_PHASE: POLYNOMIAL_TIME_DEPENDENCE, P0=phi;

RF_CAVITY: VARIABLE_RF_CAVITY, PHASE_MODEL="RF_PHASE", AMPLITUDE_MODEL="RF_AMPLITUDE",
          FREQUENCY_MODEL="RF_FREQUENCY", L=100., HEIGHT=200., WIDTH=2000.;

```

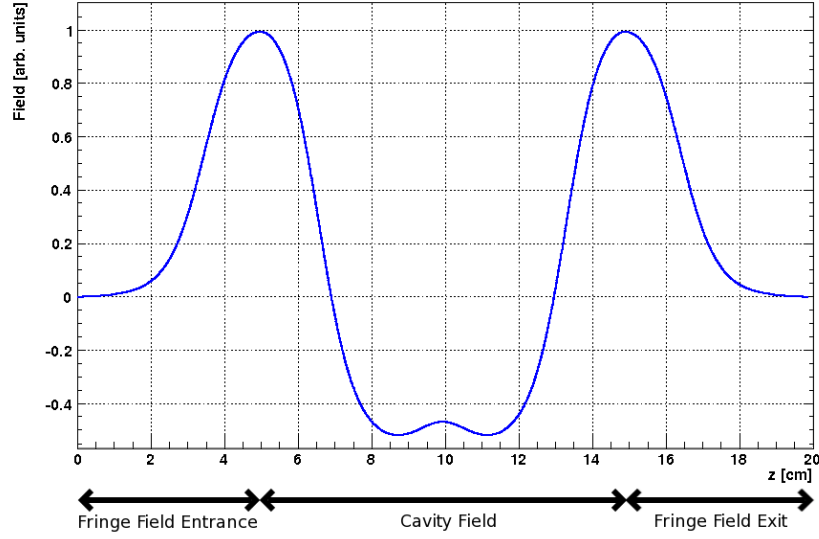


Figure 8.6: The on-axis field of an S-band (2997.924 MHz) TRAVELINGWAVE structure. The field of a single cavity is shown between its entrance and exit fringe fields. The fringe fields extend one half wavelength ($\lambda/2$) to either side.

8.14 Traveling Wave Structure

An example of a 1D TRAVELINGWAVE structure field map is shown in Figure 8.6. This map is a standing wave solution generated by Superfish and shows the field on axis for a single accelerating cavity with the fringe fields of the structure extending to either side. OPAL-T reads in this field map and constructs the total field of the TRAVELINGWAVE structure in three parts: the entrance fringe field, the structure fields and the exit fringe field.

The fringe fields are treated as standing wave structures and are given by:

$$\mathbf{E}_{\text{entrance}}(\mathbf{r}, t) = \mathbf{E}_{\text{from-map}}(\mathbf{r}) \cdot \text{VOLT} \cdot \cos(2\pi \cdot \text{FREQ} \cdot t + \phi_{\text{entrance}})$$

$$\mathbf{E}_{\text{exit}}(\mathbf{r}, t) = \mathbf{E}_{\text{from-map}}(\mathbf{r}) \cdot \text{VOLT} \cdot \cos(2\pi \cdot \text{FREQ} \cdot t + \phi_{\text{exit}})$$

where VOLT and FREQ are the field magnitude and frequency attributes (see below). $\phi_{\text{entrance}} = \text{LAG}$, the phase attribute of the element (see below). ϕ_{exit} is dependent upon both LAG and the NUMCELLS attribute (see below) and is calculated internally by OPAL-T.

The field of the main accelerating structure is reconstructed from the center section of the standing wave solution shown in Figure 8.6 using

$$\begin{aligned} \mathbf{E}(\mathbf{r}, t) = & \frac{\text{VOLT}}{\sin(2\pi \cdot \text{MODE})} \\ & \times \left\{ \mathbf{E}_{\text{from-map}}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \cdot \cos\left(2\pi \cdot \text{FREQ} \cdot t + \text{LAG} + \frac{\pi}{2} \cdot \text{MODE}\right) \right. \\ & \left. + \mathbf{E}_{\text{from-map}}(\mathbf{x}, \mathbf{y}, \mathbf{z} + \mathbf{d}) \cdot \cos\left(2\pi \cdot \text{FREQ} \cdot t + \text{LAG} + \frac{3\pi}{2} \cdot \text{MODE}\right) \right\} \end{aligned}$$

where \mathbf{d} is the cell length and is defined as $\mathbf{d} = \lambda \cdot \text{MODE}$. MODE is an attribute of the element (see below). When calculating the field from the map ($\mathbf{E}_{\text{from-map}}(\mathbf{x}, \mathbf{y}, \mathbf{z})$), the longitudinal position is referenced to the start of

the cavity fields at $\frac{\lambda}{2}$ (In this case starting at $z = 5.0$ cm). If the longitudinal position advances past the end of the cavity map ($\frac{3\lambda}{2} = 15.0$ cm in this example), an integral number of cavity wavelenths is subtracted from the position until it is back within the map's longitudinal range.

A TRAVELINGWAVE structure has seven real attributes, one integer attribute, one string attribute and one boolean attribute:

```
label:TRAVELINGWAVE, APERTURE=real-vector, L=real,
      VOLT=real, LAG=real, FMAPFN=string,
      ELEMEDGE=real, FREQ=real, NUMCELLS=integer,
      MODE=real;
```

L The length of the cavity (default: 0 m). In OPAL-T this attribute is ignored, the length is defined by the field map and the number of cells.

VOLT The peak RF voltage (default: 0 MV). The effect of the cavity is $\delta E = \text{VOLT} \cdot \sin(\text{LAG} - 2\pi \cdot \text{FREQ} \cdot t)$.

LAG The phase lag [rad] (default: 0).

FMAPFN Field maps in the *T7* format can be specified.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m. This is the position of the front edge of the first TRAVELINGWAVE cavity. (In the simulation the actual field will extend $\frac{1}{2}$ wavelength in front of this position.)

FREQ Defines the frequency of the traveling wave structure in units of MHz. A warning is issued when the frequency of the cavity card does not correspond to the frequency defined in the FMAPFN file. The frequency defined in the FMAPFN file overrides the frequency defined on the cavity card.

NUMCELLS Defines the number of cells in the tank. (The cell count should not include the entry and exit half cell fringe fields.)

MODE Defines the mode in units of 2π , for example $\frac{1}{3}$ stands for a $\frac{2\pi}{3}$ structure.

FAST If FAST is true and the provided field map is in 1D then a 2D field map is constructed from the 1D on-axis field, see section C.4. To track the particles the field values are interpolated from this map instead of using an FFT based algorithm for each particle and each step. (default: FALSE)

APVETO If TRUE this cavity will not be auto-phased, instead the actual phase on the element is used.

Use of a traveling wave requires the particle momentum **P** and the particle charge **CHARGE** to be set on the relevant optics command before any calculations are performed.

Example of a L-Band travelling wave structure:


```
lrf0: TravelingWave, L=0.0253, VOLT=14.750,
      NUMCELLS=40, ELEMEDGE=2.73066,
      FMAPFN="INLB-02-RAC.Ez", MODE=1/3,
      FREQ=1498.956, LAG=248.0/360.0;
```

8.15 Monitors

A MONITOR detects all particles passing it and writes the position, the momentum and the time when they hit it into an H5hut file. Furthermore the exact position of the monitor is stored. It has always a length of 1 cm consisting of 0.5 cm drift, the monitor of zero length and another 0.5 cm drift. This is to prevent OPAL-T from missing any particle. The positions of the particles on the monitor are interpolated from the current position and momentum one step before they would passe the monitor.

OUTFN The filename into which the monitor should write the collected data. The file is an H5hut file.

ELEMEDGE The position of the monitor is specified absolute (floor space co-ordinates) in m. This is the position at which the data is collected.

This is a restricted feature:  OPAL-CYCL .

8.16 Collimators

Three types of collimators are defined:

ECOLLIMATOR Elliptic aperture,

RCOLLIMATOR Rectangular aperture.

CCOLLIMATOR Radial rectangular collimator in cyclotrons

```
label:ECOLLIMATOR, TYPE=string, APERTURE=real-vector,
      L=real, XSIZE=real, YSIZE=real;
label:RCOLLIMATOR, TYPE=string, APERTURE=real-vector,
      L=real, XSIZE=real, YSIZE=real;
```

Either type has three real attributes:

L The collimator length (default: 0 m).

XSIZE The horizontal half-aperture (default: unlimited).

YSIZE The vertical half-aperture (default: unlimited).

For elliptic apertures, **XSIZE** and **YSIZE** denote the half-axes respectively, for rectangular apertures they denote the half-width of the rectangle. Optically a collimator behaves like a drift space, but during tracking, it also introduces an aperture limit. The aperture is checked at the entrance. If the length is not zero, the aperture is also checked at the exit.

Example:

```
COLLIM:ECOLLIMATOR, L=0.5, XSIZE=0.01, YSIZE=0.005;
```

The reference system for a collimator is a Cartesian coordinate system.

8.16.1 OPAL-T mode

The **RCOLLIMATOR** and **CCOLLIMATOR** are not supported at the moment. A **ECOLLIMATOR** detects all particles which are outside the aperture defined by **XSIZE** and **YSIZE**. The lost particles are saved into an H5hut file defined by **OUTFN**. The **ELEMEDGE** defines the location of the collimator and **L** the length.

OUTFN The filename into which the monitor should write the collected data. The file is an H5hut file.

ELEMEDGE The position of the monitor is specified absolute (floor space co-ordinates) in m. This is the position at which the data is collected.

Example:

```
Col1:ECOLLIMATOR, L=1.0E-3, ELEMEDGE=3.0E-3, XSIZE=5.0E-4,
      YSIZE=5.0E-4, OUTFN="Coll.h5";
```

8.16.2 OPAL-CYCL mode

Only `CCOLLIMATOR` is available for OPAL-CYCL. This element is radial rectangular collimator which can be used to collimate the radial tail particles. So when a particle hit this collimator, it will be absorbed or scattered, the algorithm is based on the Monte Carlo method. Please note when a particle is scattered, it will not be recorded as the lost particle. If this particle leave the bunch, it will be removed during the integration afterwards, so as to maintain the accuracy of space charge solving.

XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

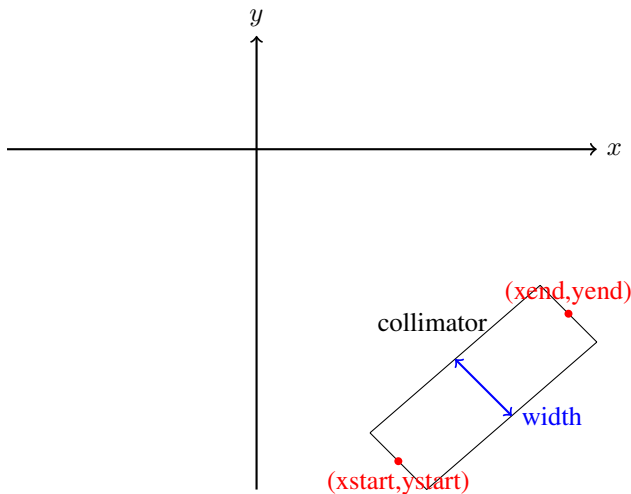
YEND The y coordinate of the end point. [mm]

ZSTART The vertical coordinate of the start point [mm]. Default value is -100 mm.

ZEND The vertical coordinate of the end point. [mm]. Default value is -100 mm.

WIDTH The width of the septum. [mm]

SURFACEPHYSICS Surfacephysics is an attribute of the element. Collimator physics is a only a kind of surfacephysics. It can be applied to any element. If the type of Surfacephysics is "Collimator", the material is defined here. The material "Cu", "Be", "Graphite" and "Mo" are defined until now. If this is not set, the surface physics module will not be activated. The particle hitting collimator will be recorded and directly deleted from the simulation.



Example:

```


y1=-0.0;
y2=0.0;
y3=200.0;
y4=205.0;
x1=-215.0;
x2=-220.0;
x3=0.0;
x4=0.0;
cmphys:surfacephysics, TYPE="Collimator", MATERIAL="Cu";
cmal: CCollimator, XSTART=x1, XEND=x2, YSTART=y1, YEND=y2,

```

```
ZSTART=2, ZEND=100, WIDTH=10.0, SURFACEPHYSICS=cmphys ;
cma2: CCollimator, XSTART=x3, XEND=x4, YSTART=y3, YEND=y4,
ZSTART=2, ZEND=100, WIDTH=10.0, SURFACEPHYSICS=cmphys;
```

The particles lost on the CCLLIMATOR are recorded in the ASCII file *inputfilename.loss*

8.17 Septum (OPAL-CYCL)

This is a restricted feature:  OPAL-T. The particles hitting on the septum is removed from the bunch. There are 5 parameters to describe a septum.

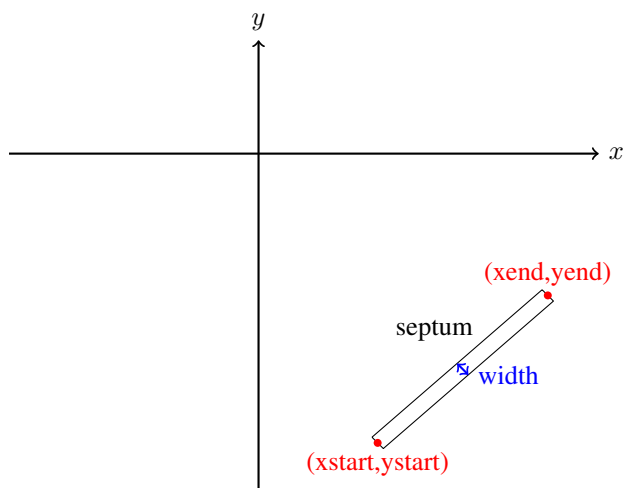
XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

YEND The y coordinate of the end point. [mm]

WIDTH The width of the septum. [mm]



Example:

```
eec2: Septum, xstart=4100.0, xend=4300.0,
ystart=-1200.0, yend=-150.0, width=0.05;
```

The particles lost on the SEPTUM are recorded in the ASCII file *inputfilename.loss*

8.18 Probe (OPAL-CYCL)

The particles hitting on the probe is recorded. There are 5 parameters to describe a probe.

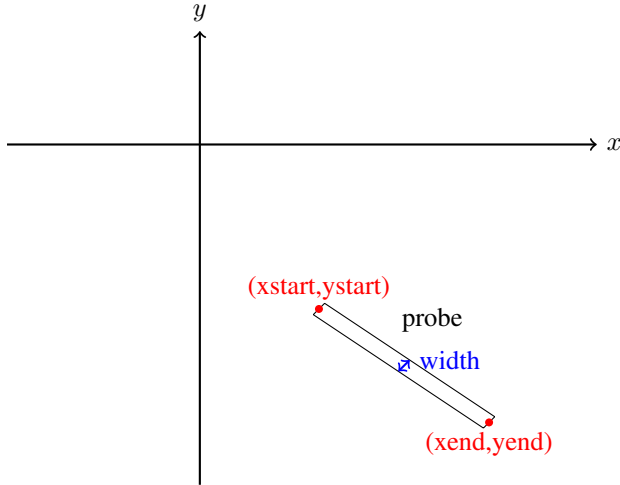
XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

YEND The y coordinate of the end point. [mm]

WIDTH The width of the probe, NOT used yet.



Example:

```
probl: Probe, xstart=4166.16, xend=4250.0,
ystart=-1226.85, yend=-1241.3;
```

The particles probed on the PROBE are recorded in the ASCII file *input.filename.loss*. Please note that these particles are not deleted in the simulation, however, they are recorded in the “loss” file.

8.19 Stripper (OPAL-CYCL)

A stripper element strip the electron(s) from a particle. The particle hitting the stripper is recorded in the file, which contains the time, coordinates and momentum of the particle at the moment it hit the stripper. The charge and mass are changed. It has the same geometry as the PROBE element. Please note that the stripping physics is not included yet.

There are 9 parameters to describe a stripper.

XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

YEND The y coordinate of the end point. [mm]

WIDTH The width of the probe, NOT used yet.

OPCHARGE Charge number of the outgoing particle. Negative value represents negative charge.

OPMASS Mass of the outgoing particles. [GeV/c²]

OPYIELD Yield of the outgoing particle (the outcome particle number per income particle), the default value is 1.

STOP If STOP is true, the particle is stopped and deleted from the simulation; Otherwise, the outgoing particle continues to be tracked along the extraction path.

Example: H_2^+ particle stripping

```

probl: Stripper, xstart=4166.16, xend=4250.0,
ystart=-1226.85, yend=-1241.3,
opcharge=1, opmass=PMASS, opyield=2, stop=false;

```

No matter what the value of STOP is, the particles hitting on the STRIPPER are recorded in the ASCII file *inputfilename.loss*.

8.20 Degrader (OPAL-T)

Rectangular degrader with an overall length L.

Example: Graphite degrader of 15 cm thickness.

```

DEGPHYS: SURFACEPHYSICS, TYPE="DEGRADER", MATERIAL="Graphite";

DEG1: DEGRADER, L=0.15, ELEMEDGE=0.02, OUTFN="DEG1.h5", SURFACEPHYSICS=DEGPHYS;

```

8.21 Correctors (OPAL-T)

Three types of closed orbit correctors are available:

HKICKER A corrector for the horizontal plane,

VKICKER A corrector for the vertical plane,

KICKER A corrector for both planes.

```

label:HKICKER, TYPE=string, APERTURE=real-vector,
      L=real, KICK=real;
label:VKICKER, TYPE=string, APERTURE=real-vector,
      L=real, KICK=real;
label:KICKER, TYPE=string, APERTURE=real-vector,
      L=real, HKICK=real, VKICK=real;

```

They have the following attributes:

L The length of the closed orbit corrector (default: 0 m).

KICK The kick angle for either horizontal or vertical correctors. (default: 0 rad).

HKICK The horizontal kick angle for a corrector in both planes (default: 0 rad).

VKICK The vertical kick angle for a corrector in both planes (default: 0 rad).

A positive kick increases p_x or p_y respectively.

Examples:

```

HK1:HKICKER, KICK=0.001;
VK3:VKICKER, KICK=0.0005;
KHV:KICKER, HKICK=0.001, VKICK=0.0005;

```

The reference system for an orbit corrector is a cartesian coordinate system.

Chapter 9

Beam Lines

The accelerator to be studied is known to OPAL as a sequence of physical elements called a **beam line**. A beam line is built from simpler beam lines whose definitions can be nested to any level. A powerful syntax allows to repeat or to reflect pieces of beam lines. Formally a beam line is defined by a `LINE` command:

```
label:LINE=(member,...,member);
```

`label` (see §6.2) gives a name to the beam line for later reference.

Each `member` may be one of the following:

- An element label,
- A beam line label,
- A sub-line, enclosed in parentheses,

Beam lines can be nested to any level.

9.1 Simple Beam Lines

The simplest beam line consists of single elements:

```
label:LINE=(member,...,member);
```

Example:

```
L:LINE=(A,B,C,D,A,D);
```

9.2 Sub-lines

Instead of referring to an element, a beam line member can refer to another beam line defined in a separate command. This provides a shorthand notation for sub-lines which occur several times in a beam line. Lines and sub-lines can be entered in any order, but when a line is used, all its sub-lines must be known.

Example:

```
L:LINE=(A,B,S,B,A,S,A,B);  
S:LINE=(C,D,E);
```

This example produces the following expansion steps:

1. Replace sub-line S:

$$(A, B, (C, D, E), B, A, (C, D, E), A, B)$$

2. Omit parentheses:

$$A, B, C, D, E, B, A, C, D, E, A, B$$

valuated to constants immediately.

Chapter 10

Physics Commands

10.1 BEAM Command

All OPAL commands working on a beam require the setting of various quantities related to this beam. These are entered by a BEAM command:

```
label:BEAM, PARTICLE=name, MASS=real, CHARGE=real,  
          ENERGY=real, PC=real, GAMMA=real, BCURRENT=real,  
          NPART=real, BUNCHED=logical, BFREQ=real;
```

The `label` is optional, it defaults to `UNNAMED_BEAM`. . The particle mass and charge are defined by:

PARTICLE The name of particles in the machine. OPAL knows the mass and the charge for the following particles:

POSITRON The particles are positrons ($MASS=m_e$, $CHARGE=1$),

ELECTRON The particles are electrons ($MASS=m_e$, $CHARGE=-1$),

PROTON The particles are protons (default, $MASS=m_p$, $CHARGE=1$),

ANTIPROTON The particles are anti-protons ($MASS=m_p$, $CHARGE=-1$).

HMINUS The particles are h- protons ($MASS=m_{h^-}$, $CHARGE=-1$).

CARBON The particles are carbons ($MASS=m_c$, $CHARGE=12$).

URANIUM The particles are of type uranium ($MASS=m_u$, $CHARGE=35$).

MUON The particles are of type muon ($MASS=m_\mu$, $CHARGE=-1$).

DEUTERON The particles are of type deuteron ($MASS=m_d$, $CHARGE=1$).

XENON The particles are of type xenon ($MASS=m_{xe}$, $CHARGE=20$).

For other particle names one may enter:

MASS The particle mass in GeV.

CHARGE The particle charge expressed in elementary charges.

The other attributes are:

BFREQ The bunch frequency in MHz.

BCURRENT The bunch current in A.

$$\text{BCURRENT} = Q \times \text{BFREQ} \quad (10.1)$$

with Q the total charge.

NPART The number of macro particles for the simulations

Chapter 11

Distribution Command

Table 11.1: Possible distribution types. Note that the SURFACEEMISSION and SURFACERANDCREATE distribution types will not be discussed in this chapter. Instead, refer to Chapter 16 on field emission for using these types.

Distribution Type	Description
FROMFILE	Initial distribution read in from text file provided by user. (See section 11.3.)
GAUSS	Initial distribution generated using Gaussian distribution(s). (See section 11.4.)
FLATTOP	Initial distribution generated using flattop distribution(s). (See section 11.5.)
BINOMIAL	Initial distribution generated using binomial distribution(s). (See section 11.6.)
SURFACEEMISSION	For dark current and multipacting simulations. See Chapter 16. This type of distribution will not be covered in this chapter.
SURFACERANDCREATE	For dark current and multipacting simulations. See Chapter 16. This type of distribution will not be covered in this chapter.
GUNGAUSSFLATTOPTH	Legacy. Special case of FLATTOP distribution. (See 11.5.4.)
ASTRAFLATTOPTH	Legacy. Special case of FLATTOP distribution. (See 11.5.5.)

The distribution command is used to introduce particles into an OPAL simulation. Like other OPAL commands (Chapter 6), the distribution command is of the form:

```
Name:DISTRIBUTION, DISTRIBUTION = TYPE,
      ATTRIBUTE #1 =,
      ATTRIBUTE #2 =,
      .
      .
      .
      ATTRIBUTE #N =;
```

The distribution is given a name (which is used to reference the distribution in the OPAL input file), a distribution type, and a list of attributes. The types of distributions that are supported are listed in Table 11.1. The attributes that follow the distribution type further define the particle distribution. Some attributes are universal, while others are specific to the distribution type. In the following sections we will define the distribution attributes, starting with the general, or universal attributes. (Note that, in general, if a distribution type does not support a particular attribute, defining a value for it does no harm. That attribute just gets ignored.)

11.1 Units

The internal units used by OPAL-T and OPAL-CYCL are described in Sections 4.2 and 5.3. When defining a distribution, both OPAL-T and OPAL-CYCL use meters for length and seconds for time. However, there are different options for the units used to input the momentum. This is controlled with the INPUTMOUNITS attribute, defined in Table 11.2.

Table 11.2: Definition of INPUTMOUNITS attribute.

Attribute Name	Value	Description
INPUTMOUNITS	NONE (default for OPAL-T)	Use no units for the input momentum (e.g. p_x , p_y , p_z). Momentum is given as $\beta_x\gamma$, $\beta_y\gamma$ and $\beta_z\gamma$, as in Section 4.2.
INPUTMOUNITS	EV (default for OPAL-CYCL)	Use the units eV for the input momentum (e.g. p_x , p_y , p_z).

11.2 General Distribution Attributes

Once the distribution type is chosen, the next attribute to specify is the EMITTED attribute, which is defined in Table 11.3. The EMITTED attribute controls whether a distribution is *injected* or *emitted*. An *injected* distribution is placed in its entirety into the simulation space at the start of the simulation. An *emitted* beam is emitted into the simulation over time as the simulation progresses (e.g. from a cathode in a photoinjector simulation). Currently, only OPAL-T supports *emitted* distributions. The default is an *injected* distribution.

Depending on the EMITTED attribute, we can specify several other attributes that do not depend on the distribution type. These are defined in the Subsections 11.2.1, 11.2.2 and 11.2.3 in Tables 11.4, 11.5 and 11.6.

Table 11.3: Definition of EMITTED attribute.

Attribute Name	Value	Description
EMITTED	FALSE (default)	<p>The distribution is injected into the simulation in its entirety at the start of the simulation.</p> <p>The particle coordinates for an injected distribution are defined as in Sections 4.2 and 5.3.</p> <p>Note that in OPAL-T the entire distribution will automatically be shifted to ensure that the z coordinate will be greater than zero for all particles.</p>
EMITTED	TRUE	<p>The distribution is emitted into the simulation over time as the simulation progresses.</p> <p>Currently only OPAL-T supports this type of distribution. In this case, the longitudinal coordinate, as defined by Section 4.2, is given in seconds instead of meters. Early times are emitted first.</p>

11.2.1 Universal Attributes

Table 11.4: Definition of universal distribution attributes. Any distribution type can use these and they are the same whether the beam is *injected* or *emitted*.

Attribute Name	Default Value	Units	Description
WRITETOFILE	FALSE	None	Echo initial distribution to text file “data/distribution.data”.
WEIGHT	1.0	None	Weight of distribution when used in a distribution list. See Section 11.8.
NBIN	0	None	The distribution (beam) will be broken up into NBIN energy bins. This has consequences for the space charge solver. (See Section 12.14.)
XMULT	1.0	None	Value used to scale the x positions of the distribution particles. Applied after the distribution is generated (or read in).
YMULT	1.0	None	Value used to scale the y positions of the distribution particles. Applied after the distribution is generated (or read in).
PXMULT	1.0	None	Value used to scale the x momentum, p_x , of the distribution particles. Applied after the distribution is generated (or read in).
PYMULT	1.0	None	Value used to scale the y momentum, p_y , of the distribution particles. Applied after the distribution is generated (or read in).
PZMULT	1.0	None	Value use to scale the z momentum, p_z , of the distribution particles. Applied after the distribution is generated (or read in).
OFFSETX	0.0	m	Distribution is shifted in x by this amount after the distribution is generated (or read in). Same as the average x position, \bar{x} .
OFFSETY	0.0	m	Distribution is shifted in y by this amount after the distribution is generated (or read in). Same as the average y position, \bar{y} .
OFFSETPX	0.0	Section 11.1	Distribution is shifted in p_x by this amount after the distribution is generated (or read in). Same as the average p_x value, \bar{p}_x .
OFFSETPY	0.0	Section 11.1	Distribution is shifted in p_y by this amount after the distribution is generated (or read in). Same as the average p_y value, \bar{p}_y .
OFFSETPZ	0.0	Section 11.1	Distribution is shifted in p_z by this amount after the distribution is generated (or read in). Same as the average p_z value, \bar{p}_z .

11.2.2 Injected Distribution Attributes

Table 11.5: Definition of distribution attributes that only affect *injected* beams.

Attribute Name	Default Value	Units	Description
ZMULT	1.0	None	Value used to scale the z positions of the distribution particles. Applied after the distribution is generated (or read in).
OFFSETZ	0.0	m	Distribution is shifted in z by this amount after the distribution is generated (or read in). Same as the average z position, \bar{z} .

11.2.3 Emitted Distribution Attributes

Table 11.6: Definition of distribution attributes that only affect *emitted* beams.

Attribute Name	Default Value	Units	Description
TMULT	1.0	None	Value used to scale the t values of the distribution particles. Applied after the distribution is generated (or read in).
OFFSETT	0.0	s	Distribution is shifted in t by this amount after the distribution is generated (or read in). Note that this attribute only has an effect in a distribution list. (See Section 11.8).
EMISSIONSTEPS	1	None	Number of time steps to take during emission. The simulation time step will be adjusted during emission to ensure that this many time steps will be required to emit the entire distribution.
EMISSIONMODEL	None	None	Emission model to use when emitting particles from cathode. (See Section 11.7).

11.3 FROMFILE Distribution Type

The most versatile distribution type is to use a user generated text file as input to OPAL. This allows the user to generate their own distribution, if the built in options in OPAL are insufficient, and have it either *injected* or *emitted* into the simulation. In Table 11.7 we list the single attribute specific to this type of distribution type.

Table 11.7: Definition of distribution attributes for a FROMFILE distribution type.

Attribute Name	Default Value	Units	Description
FNAME		None	File name for text file containing distribution particle coordinates.

An example of an *injected* FROMFILE distribution definition is:

```
Name:DISTRIBUTION, DISTRIBUTION = FROMFILE,
                        FNAME = ``text file name``;
```

an example of an *emitted* FROMFILE distribution definition is:

```
Name:DISTRIBUTION, DISTRIBUTION = FROMFILE,
                        FNAME = ``text file name``,
                        EMITTED = TRUE,
                        EMISSIONMODEL = None;
```

The text input file for the FROMFILE distribution type has slightly a slightly different format, depending on whether the distribution is to be *injected* or *emitted*. The *injected* file format is defined in Table 11.8. The *emitted* file format is defined in Table 11.9.

Table 11.8: File format for *injected* FROMFILE distribution type. N is the number of particles in the file. The particle coordinates are defined in Sections 4.2 or 5.3.

N					
x_1	p_{x1}	y_1	p_{y1}	z_1	p_{z1}
x_2	p_{x2}	y_2	p_{y2}	z_2	p_{z2}
.					
.					
x_N	p_{xN}	y_N	p_{yN}	z_N	p_{zN}

Note that for an *emitted* FROMFILE distribution, all of the particle's time, t , coordinates will be shifted to negative time (if they are not there already). The simulation clock will then start at $t = 0$ and distribution particles will be emitted into the simulation as the simulation progresses. Also note that, as the particles are emitted, they will be modified according to the type of emission model used. This is defined by the attribute EMISSIONMODEL, which is described in Section 11.7. A choice of NONE for the EMISSIONMODEL (which is the default) can be defined so as not to affect the distribution coordinates at all.

To maintain consistency N and NPART from the BEAM command 10.1 must be equal.

Table 11.9: File format for *emitted* FROMFILE distribution type. N is the number of particles in the file. The particle coordinates are defined in Section 4.2 except that z , in meters, is replaced by t in seconds.

N					
x_1	p_{x1}	y_1	p_{y1}	t_1	p_{z1}
x_2	p_{x2}	y_2	p_{y2}	t_2	p_{z2}
.					
.					
x_N	p_{xN}	y_N	p_{yN}	t_N	p_{zN}

11.4 GAUSS Distribution Type

As the name implies, the GAUSS distribution type can generate distributions with a general Gaussian shape (here we show a one dimensional example):

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma_x^2}}$$

where \bar{x} is the average value of x . However, the GAUSS distribution can also be used to generate an emitted beam with a flat top time profile. We will go over the various options for the GAUSS distribution type in this section.

11.4.1 Simple GAUSS Distribution Type

We will begin by describing how to create a simple GAUSS distribution type. That is, a simple 6 dimensional distribution with a Gaussian distribution in all dimensions.

Table 11.10: Definition of the basic distribution attributes for a GAUSS distribution type.

Attribute Name	Default Value	Units	Description
SIGMAX	0.0	m	RMS width, σ_x , in transverse x direction.
SIGMAY	0.0	m	RMS width, σ_y , in transverse y direction.
SIGMAR	0.0	m	RMS radius, σ_r , in radial direction. If nonzero SIGMAR overrides SIGMAX and SIGMAY.
SIGMAZ	0.0	m	RMS length, σ_z , in longitudinal (z) direction. SIGMAZ is used for an <i>injected</i> distribution.
SIGMAT	0.0	s	RMS width, σ_t , in time (t). SIGMAT is used for an <i>emitted</i> distribution.
SIGMAPX	0.0	Section 11.1	Parameter σ_{px} for defining distribution.
SIGMAPY	0.0	Section 11.1	Parameter σ_{py} for defining distribution.
SIGMAPZ	0.0	Section 11.1	Parameter σ_{pz} for defining distribution.
CUTOFFX	3.0	None	Defines transverse distribution cutoff in the x direction in units of σ_x . If CUTOFFX = 0 then actual cutoff in x is set to infinity.
CUTOFFY	3.0	None	Defines transverse distribution cutoff in the y direction in units of σ_y . If CUTOFFY = 0 then actual cutoff in y is set to infinity.
CUTOFFR	3.0	None	Defines transverse distribution cutoff in the r direction in units of σ_r . If CUTOFFR = 0 then actual cutoff in r is set to infinity. CUTOFFR is only used if SIGMAR > 0.
CUTOFFLONG	3.0	None	Defines longitudinal distribution cutoff in the z or t direction (<i>injected</i> or <i>emitted</i>) in units of σ_z or σ_t . CUTOFFLONG is different from other dimensions in that a value of 0.0 does not imply a cutoff value of infinity.
CUTOFFPX	3.0	None	Defines cutoff in p_x dimension in units of σ_{px} . If CUTOFFPX = 0 then actual cutoff in p_x is set to infinity.
CUTOFFPY	3.0	None	Defines cutoff in p_y dimension in units of σ_{py} . If CUTOFFPY = 0 then actual cutoff in p_y is set to infinity.
CUTOFFPZ	3.0	None	Defines cutoff in p_z dimension in units of σ_{pz} . If CUTOFFPZ = 0 then actual cutoff in p_z is set to infinity.

In Table 11.10 we list the basic attributes available for a GAUSS distribution. We can use these to create a very simple GAUSS distribution:

```
Name:DISTRIBUTION, DISTRIBUTION = GAUSS,
      SIGMAX = 0.001,
      SIGMAY = 0.003,
      SIGMAZ = 0.002,
      SIGMAPX = 0.0,
      SIGMAPY = 0.0,
      SIGMAPZ = 0.0,
      CUTOFFX = 2.0,
      CUTOFFY = 2.0,
      CUTOFFLONG = 4.0,
      OFFSETX = 0.001,
      OFFSETY = -0.002,
      OFFSETZ = 0.01,
      OFFSETPZ = 1200.0
      USEEV = TRUE;
```

This creates a Gaussian shaped distribution with zero transverse emittance, zero energy spread, $\sigma_x = 1.0$ mm, $\sigma_y = 3.0$ mm, $\sigma_z = 2.0$ mm and an average energy of:

$$W = 1.2 \text{ MeV}$$

In the x direction, the Gaussian distribution is cutoff at $x = 2.0 \times \sigma_x = 2.0$ mm. In the y direction it is cutoff at $y = 2.0 \times \sigma_y = 6.0$ mm. This distribution is *injected* into the simulation at an average position of $(\bar{x}, \bar{y}, \bar{z}) = (1.0 \text{ mm}, -2.0 \text{ mm}, 10.0 \text{ mm})$.

11.4.2 GAUSS Distribution for Photoinjector

Table 11.11: Definition of additional distribution attributes for an *emitted* GAUSS distribution type. These are used to generate a distribution with a time profile as illustrated in Figure 11.1.

Attribute Name	Default Value	Units	Description
TPULSEFWHM	0.0	s	Flat top time. See Figure 11.1.
TRISE	0.0	s	Rise time. See Figure 11.1. If defined will override SIGMAT.
TFALL	0.0	s	Fall time. See Figure 11.1. If defined will override SIGMAT.
FTOSAMPLITUDE	0	None	Sinusoidal oscillations can imposed on the flat top in Figure 11.1. This defines the amplitude of those oscillations in percent of the average flat top amplitude.
FTOSPERIODS	0	None	Defines the number of oscillation periods imposed on the flat top, t_{flattop} , in Figure 11.1.

A useful feature of the GAUSS distribution type is the ability to mimic the initial distribution from a photoinjector. For this purpose we have the distribution attributes listed in Table 11.11. Using them, we can create a distribution with the time structure shown in Figure 11.1. This is a half Gaussian rise plus a uniform flat-top plus a half

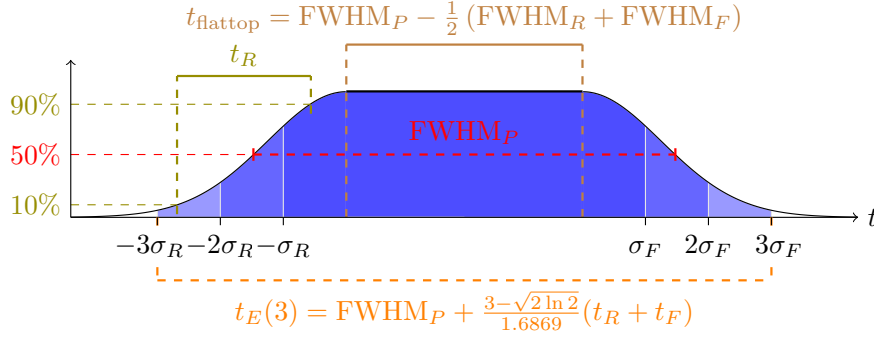


Figure 11.1: OPAL emitted GAUSS distribution with flat top.

Gaussian fall. To make it more convenient to mimic measured laser profiles, TRISE and TFALL from Table 11.11 do not define RMS quantities, but instead are given by (See also Figure 11.1):

$$\begin{aligned}
 \text{TRISE} = t_R &= \left(\sqrt{2 \ln(10)} - \sqrt{2 \ln\left(\frac{10}{9}\right)} \right) \sigma_R \\
 &= 1.6869 \sigma_R \\
 \text{TFALL} = t_F &= \left(\sqrt{2 \ln(10)} - \sqrt{2 \ln\left(\frac{10}{9}\right)} \right) \sigma_F \\
 &= 1.6869 \sigma_F
 \end{aligned}$$

where σ_R and σ_F are the Gaussian, RMS rise and fall times respectively. The flat-top portion of the profile, TPULSEFWHM, is defined as (See also Figure 11.1):

$$\text{TPULSEFWHM} = \text{FWHM}_P = t_{\text{flattop}} + \sqrt{2 \ln 2} (\sigma_R + \sigma_F)$$

Total emission time, t_E , of this distribution, is a function of the longitudinal cutoff, CUTOFFLONG (See Table 11.10), and is given by:

$$\begin{aligned}
 t_E(\text{CUTOFFLONG}) &= \text{FWHM}_P - \frac{1}{2} (\text{FWHM}_R + \text{FWHM}_F) + \text{CUTOFFLONG} (\sigma_R + \sigma_F) \\
 &= \text{FWHM}_P + \frac{\text{CUTOFFLONG} - \sqrt{2 \ln 2}}{1.6869} (\text{TRISE} + \text{TFALL})
 \end{aligned}$$

Finally, we can also impose oscillations over the flat-top portion of the laser pulse in Figure 11.1, t_{flattop} . This is defined by the attributes FTOSCAMPLITUDE and FTOSCPERIODS from Table 11.11. FTOSCPERIODS defines how many oscillation periods will be present during the t_{flattop} portion of the pulse. FTOSCAMPLITUDE defines the amplitude of those oscillations in percentage of the average profile amplitude during t_{flattop} . So, for example, if we set FTOSCAMPLITUDE = 5, and the amplitude of the profile is equal to 1.0 during t_{flattop} , the amplitude of the oscillation will be 0.05.

11.4.3 Correlations for GAUSS Distribution (Experimental)

Table 11.12: Definition of additional distribution attributes for a GAUSS distribution type for generating correlations in the beam.

Attribute Name	Default Value	Units	Description
CORRX	0.0	Section 11.1	x, p_x correlation. (R_{12} in transport notation.)
CORRY	0.0	Section 11.1	y, p_y correlation. (R_{34} in transport notation.)
CORRZ	0.0	Section 11.1	z, p_z correlation. (R_{56} in transport notation.)
R51	0.0	Section 11.1	x, z correlation. (R_{51} in transport notation.)
R52	0.0	Section 11.1	p_x, z correlation. (R_{52} in transport notation.)
R61	0.0	Section 11.1	x, p_z correlation. (R_{61} in transport notation.)
R62	0.0	Section 11.1	p_x, p_z correlation. (R_{62} in transport notation.)

To generate Gaussian initial distribution with dispersion, first we generate the uncorrelated gaussian inputs matrix $R = (R_1, \dots, R_n)$. The mean of R_i is 0 and the standard deviation squared is 1. Then we correlate R . The correlation coefficient matrix σ in x, p_x, z, p_z phase space reads:

$$\sigma = \begin{bmatrix} 1 & c_x & r51 & r61 \\ c_x & 1 & r52 & r62 \\ r51 & r52 & 1 & c_t \\ r61 & r62 & c_t & 1 \end{bmatrix}$$

The Cholesky decomposition of the symmetric positive-definite matrix σ is $\sigma = C^T C$, then the correlated distribution is $C^T R$.

Note: Correlations work for the moment only with the Gaussian distribution and are experimental, so there are no guarantees as to its efficacy or accuracy. Also, these correlations will work, in principle, for an *emitted* beam. However, recall that in this case, z in meters is replaced by t in seconds, so take care.

As an example of defining a correlated beam, let the initial correlation coefficient matrix be:

$$\sigma = \begin{bmatrix} 1 & 0.756 & 0.023 & 0.496 \\ 0.756 & 1 & 0.385 & -0.042 \\ 0.023 & 0.385 & 1 & -0.834 \\ 0.496 & -0.042 & -0.834 & 1 \end{bmatrix}$$

then the corresponding distribution command will read:

```

Dist:DISTRIBUTION, DISTRIBUTION = GAUSS,
                    SIGMAX = 4.796e-03,
                    SIGMAPX = 231.0585,
                    CORRX = 0.756,
                    SIGMAY = 23.821e-03,
                    SIGMAPY = 1.6592e+03,
                    CORRY = -0.999,
                    SIGMAZ = 0.466e-02,
                    SIGMAPZ = 74.7,
                    CORRZ = -0.834,
                    OFFSETZ = 0.466e-02,
                    OFFSETPZ = 72e6,
                    R61 = 0.496,
                    R62 = -0.042,
                    R51 = 0.023,
                    R52 = 0.385;

```

11.5 FLATTOP Distribution Type

The `FLATTOP` distribution type is used to define hard edge beam distributions. Hard edge, in this case, means a more or less uniformly filled cylinder of charge, although as we will see this is not always the case. The main purpose of the `FLATTOP` is to mimic laser pulses in photoinjectors, and so we usually will make this an *emitted* distribution. However it can be *injected* as well.

11.5.1 Injected FLATTOP

The attributes for an *injected* `FLATTOP` distribution are defined in Tables 11.13 and 11.4. At the moment, we cannot define a spread in the beam momentum, so an *injected* `FLATTOP` distribution will currently have zero emittance. An *injected* `FLATTOP` will be a uniformly filled ellipse transversely with a uniform distribution in z . (Basically a cylinder with an elliptical cross section.)

Table 11.13: Definition of the basic distribution attributes for an *injected* `FLATTOP` distribution type.

Attribute Name	Default Value	Units	Description
SIGMAX	0.0	m	Hard edge width in x direction.
SIGMAY	0.0	m	Hard edge width in y direction.
SIGMAR	0.0	m	Hard edge radius. If nonzero SIGMAR overrides SIGMAX and SIGMAY.
SIGMAZ	0.0	m	Hard edge length in z direction.

11.5.2 Emitted FLATTOP

Table 11.14: Definition of the basic distribution attributes for an *emitted* FLATTOP distribution type.

Attribute Name	Default Value	Units	Description
SIGMAX	0.0	m	Hard edge width in x direction.
SIGMAY	0.0	m	Hard edge width in y direction.
SIGMAR	0.0	m	Hard edge radius. If nonzero SIGMAR overrides SIGMAX and SIGMAY.
SIGMAT	0.0	s	RMS rise and fall of half Gaussians in flat top defined in in Figure 11.1.
TPULSEFWHM	0.0	s	Flat top time. See Figure 11.1.
TRISE	0.0	s	Rise time. See Figure 11.1. If defined will override SIGMAT.
TFALL	0.0	s	Fall time. See Figure 11.1. If defined will override SIGMAT.
FTOSCAMPLITUDE	0	None	Sinusoidal oscillations can imposed on the flat top in Figure 11.1. This defines the amplitude of those oscillations in percent of the average flat top amplitude.
FTOSCPERIODS	0	None	Defines the number of oscillation periods imposed on the flat top, t_{flattop} , in Figure 11.1.
LASERPROFFN		None	File name containing measured laser image.
IMAGENAME		None	Name of the file containing the laser image.
INTENSITYCUT	0.0	None	Parameter defining floor of the background to be subtracted from the laser image in percent of the maximum intensity.

The attributes of an *emitted* FLATTOP distribution are defined in Tables 11.14 and 11.4. The FLATTOP distribution was really intended for this mode of operation in order to mimic common laser pulses in photoinjectors. The basic characteristic of a FLATTOP is a uniform, elliptical transverse distribution and a longitudinal (time) distribution with a Gaussian rise and fall time as described in Section 11.4.2. Below we show an example of a FLATTOP distribution command with an elliptical cross section of 1 mm by 2 mm and a flat top, in time, 10 ps long with a 0.5 ps rise and fall time as defined in Figure 11.1.

```

Dist:DISTRIBUTION, DISTRIBUTION = FLATTOP,
    SIGMAX = 0.001,
    SIGMAY = 0.002,
    TRISE = 0.5e-12,
    TFALL = 0.5e-12,
    TPULSEFWHM = 10.0e-12,
    CUTOFFLONG = 4.0,
    NBIN = 5,
    EMISSIONSTEPS = 100,
    EMISSIONMODEL = ASTRA,

```



```
EKIN = 0.5,
EMITTED = TRUE;
```

11.5.3 Transverse Distribution from Laser Profile (Under Development)

An alternative to using a uniform, elliptical transverse profile is to define the `LASERPRFFN`, `IMAGENAME` and `INTENSITYCUT` attributes from Table 11.14. Then, OPAL-T will use the laser image as the basis to sample the transverse distribution.

This distribution option is not yet available.

11.5.4 GUNGAUSSFLATTOPTH Distribution Type

This is a legacy distribution type. A `GUNGAUSSFLATTOPTH` is the equivalent of a `FLATTOP` distribution, except that the `EMITTED` attribute will set to `TRUE` automatically and the `EMISSIONMODEL` will be automatically set to `ASTRA`.

11.5.5 ASTRAFLATTOPTH Distribution Type

This is a legacy distribution type. A `ASTRAFLATTOPTH` is the equivalent of a `FLATTOP` distribution, except that the `EMITTED` attribute will set to `TRUE` automatically and the `EMISSIONMODEL` will be automatically set to `ASTRA`. There are a few other differences with how the longitudinal time profile of the distribution is generated.

11.6 BINOMIAL Distribution Type

The `BINOMIAL` type of distribution is based on [42]. The shape of the binomial distribution is governed by one parameter m . By varying this single parameter one obtains the most commonly used distributions for our type of simulations, as listed in Table 11.15.

Table 11.15: Different distributions specified by a single parameter m

m	Distribution	Density	Profile
0.0	Hollow shell	$\frac{1}{\pi} \delta(1 - r^2)$	$\frac{1}{\pi} (1 - r^2)^{-0.5}$
0.5	Flat profile	$\frac{1}{2\pi} (1 - r^2)^{-0.5}$	$\frac{1}{2}$
1.0	Uniform	$\frac{1}{\pi}$	$\frac{2}{\pi} (1 - x^2)^{0.5}$
1.5	Elliptical	$\frac{3}{2\pi} (1 - r^2)^{0.5}$	$\frac{1}{4} (1 - x^2)$
2.0	Parabolic	$\frac{2}{\pi} (1 - r^2)$	$\frac{3}{8\pi} (1 - x^2)^{1.5}$
$\rightarrow \infty$	Gaussian	$\frac{1}{2\pi\sigma_x\sigma_y} \exp(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2})$	$\frac{1}{\sqrt{2\pi}\sigma_x} \exp(-\frac{x^2}{2\sigma_x^2})$

11.7 Emission Models

When emitting a distribution from a cathode, there are several ways in which we can model the emission process in order to calculate the thermal emittance of the beam. In this section we discuss the various options available.

11.7.1 Emission Model: NONE (default)

The emission model `NONE` is the default emission model used in OPAL-T. It has a single attribute, listed in Table 11.16. The `NONE` emission model is very simplistic. It merely adds the amount of energy defined by the attribute `EKIN` to the longitudinal momentum, p_z , for each particle in the distribution as it leaves the cathode.

Table 11.16: Attributes for the `NONE` and `ASTRA` emission models.

Attribute Name	Default Value	Units	Description
<code>EKIN</code>	1.0	eV	Thermal energy added to beam during emission.

An example of using the `NONE` emission model is given below. This option allows us to emit transversely cold (zero x and y emittance) beams into our simulation. We must add some z momentum to ensure that the particles drift into the simulation space. If in this example one were to specify `EKIN = 0`, then you would likely get strange results as the particles would not move off the cathode, causing all of the emitted charge to pile up at $z = 0$ in the first half time step before the beam space charge is calculated.

```

Dist:DISTRIBUTION, DISTRIBUTION = FLATTOP,
                      SIGMAX = 0.001,
                      SIGMAY = 0.002,
                      TRISE = 0.5e-12,
                      TFALL = 0.5e-12,
                      TPULSEFWHM = 10.0e-12,
                      CUTOFFLONG = 4.0,
                      NBIN = 5,
                      EMISSIONSTEPS = 100,
                      EMISSIONMODEL = NONE,
                      EKIN = 0.5,
                      EMITTED = TRUE;

```

One thing to note, it may be that if you are emitting your own distribution using the `DISTRIBUTION = FROMFILE` option, you may want to set `EKIN = 0` if you have already added some amount of momentum, p_z , to the particles.

11.7.2 Emission Model: ASTRA

The `ASTRA` emittance model uses the same single parameter as the `NONE` option as listed in Table 11.16. However, in this case, the energy defined by the `EKIN` attribute is added to each emitted particle's momentum in a random way:

$$\begin{aligned}
 p_{total} &= \sqrt{\left(\frac{EKIN}{mc^2} + 1\right)^2 - 1} \\
 p_x &= p_{total} \sin(\phi) \cos(\theta) \\
 p_y &= p_{total} \sin(\phi) \sin(\theta) \\
 p_z &= p_{total} |\cos(\theta)|
 \end{aligned}$$

where θ is a random angle between 0 and π , and ϕ is given by

$$\phi = 2.0 \arccos(\sqrt{x})$$

with x a random number between 0 and 1.

11.7.3 Emission Model: NONEQUIL

The NONEQUIL emission model is based on an actual physical model of particle emission as described in [49, 50, 51]. The attributes needed by this emission model are listed in Table 11.17.

Table 11.17: Attributes for the NONE and ASTRA emission models.

Attribute Name	Default Value	Units	Description
ELASER	4.86	eV	Photoinjector drive laser energy. (Default is 255 nm light.)
W	4.31	eV	Photocathode work function. (Default is atomically clean copper.)
FE	7.0	eV	Fermi energy of photocathode. (Default is atomically clean copper.)
CATHTEMP	300.0	K	Operating temperature of photocathode.

An example of using the NONEQUIL emission model is given below. This model is relevant for metal cathodes and cathodes such as *CsTe*.

```
Dist:DISTRIBUTION, DISTRIBUTION = GAUSS,
                    SIGMAX = 0.001,
                    SIGMAY = 0.002,
                    TRISE = 1.0e-12,
                    TFALL = 1.0e-12,
                    TPULSEFWHM = 15.0e-12,
                    CUTOFFLONG = 3.0,
                    NBIN = 10,
                    EMISSIONSTEPS = 100,
                    EMISSIONMODEL = NONEQUIL,
                    ELASER = 6.48,
                    W = 4.1,
                    FE = 7.0,
                    CATHTEMP = 325,
                    EMITTED = TRUE;
```

11.8 Distribution List

It is possible to use multiple distributions in the same simulation. We do this by using a distribution list in the RUN command (See Chapter 15). Assume we have defined several distributions: DIST1, DIST2 and DIST3. If we want to use just one of these distributions in a simulation, we would use the following RUN command to start the simulation:

```
RUN, METHOD = "PARALLEL-T",
      BEAM = beam_name,
      FIELDSOLVER = field_solver_name,
      DISTRIBUTION = DIST1;
```

If we want to use all the distributions at the same time, then the command would instead be:

```
RUN, METHOD = "PARALLEL-T",
      BEAM = beam_name,
      FIELDSOLVER = field_solver_name,
      DISTRIBUTION = {DIST1, DIST2, DIST3};
```

In this second case, the first distribution (DIST1) is the master distribution. The main consequence of this is that all distributions in the list will be forced to the same EMITTED condition as DIST1. So, if DIST1 is to be *emitted*, then all other distributions in the list will be forced to this same condition. If DIST1 is to be *injected*, then all other distributions in the list will also be *injected*.

The number of particles in the simulation is defined in the BEAM command (See Section 10.1). The number of particles in each distribution in a distribution list is determined by this number and the WEIGHT attribute of each distribution (Table 11.4). If all distributions have the same WEIGHT value, then the number of particles will be divided up evenly among them. If, however we have a distribution list consisting of two distributions, and one has twice the WEIGHT of the other, then it will have twice the particles as its partner. The exception here is any FROMFILE distribution type. In this case, the WEIGHT attribute and the number of particles in the BEAM command are ignored. The number of particles in any FROMFILE distribution type is defined by the text file containing the distribution particle coordinates. (Section 11.3).

Chapter 12

Fieldsolver

[TODO: AA will rewrite]

12.1 Fieldsolver Command

See Table 12.1 for a summary of the Fieldsolver command.

12.2 Define the Fieldsolver to be used

At present only a FFT based solver is available. Future solvers will include Finite Element solvers and a Multi Grid solver with Shortley-Weller boundary conditions for irregular domains.

12.3 Define Domain Decomposition

The dimensions in x , y and z can be parallel (TRUE) or serial FALSE. The default settings are: parallel in z and serial in x and y .

12.4 Define Number of Gridpoints

Number of grid points in x , y and z for a rectangular grid.

12.5 Define Boundary Conditions

Two boundary conditions can be selected independently among x , y namely: OPEN and for z OPEN & PERIODIC. In the case you select for z periodic you are about to model a DC-beam.

12.6 Define Greens Function

Two Greens functions can be selected: INTEGRATED, STANDARD. The integrated Green's function is described in [39, 40, 41]. Default setting is INTEGRATED.

Table 12.1: Fieldsolver command summary

Command	Purpose
<code>FIELDSOLVER</code> Specify a fieldsolver	
<code>FSTYPE</code>	Specify the type of field solver: FFT, FFTPERIODIC, MG, AMR and NONE
<code>PARFFTX</code>	If TRUE, the dimension x is distributed among the processors
<code>PARFFTY</code>	If TRUE, the dimension y is distributed among the processors
<code>PARFFTZ</code>	If TRUE, the dimension z is distributed among the processors
<code>MX</code>	Number of grid points in x specifying rectangular grid
<code>MY</code>	Number of grid points in y specifying rectangular grid
<code>MZ</code>	Number of grid points in z specifying rectangular grid
<code>BCFFTX</code>	Boundary condition in x [OPEN]
<code>BCFFTY</code>	Boundary condition in y [OPEN]
<code>BCFFTZ</code>	Boundary condition in z [OPEN,PERIODIC]
<code>GREENSF</code>	Defines the Greens function for the FFT Solver
<code>BBOXINCR</code>	Enlargement of the bounding box in %
<code>GEOMETRY</code>	Geometry to be used as domain boundary
<code>ITSOLVER</code>	Type of iterative solver
<code>INTERPL</code>	Interpolation used for boundary points
<code>TOL</code>	Tolerance for iterative solver
<code>MAXITERS</code>	Maximum number of iterations of iterative solver
<code>PRECMODE</code>	Behaviour of the preconditioner

12.7 Define Bounding Box Enlargement

The bounding box defines a minimal rectangular domain including all particles. With `BBOXINCR` the bounding box can be enlarged by a factor given in percent of the minimal rectangular domain.

12.8 Define Geometry

The list of geometries defining the beam line boundary. For further details see Chapter 14.

12.9 Define Iterative Solver

The iterative solver for solving the preconditioned system: CG, BiCGSTAB or GMRES.

12.10 Define Interpolation for Boundary Points

The interpolation method for gridpoints near the boundary: CONSTANT, LINEAR or QUADRATIC.

12.11 Define Tolerance

The tolerance for the iterative solver used by the MG solver.

12.12 Define Maximal Iterations

The maximal number of iterations the iterative solver performs.

12.13 Define Preconditioner Behaviour

The behaviour of the preconditioner can be: STD, HIERARCHY or REUSE. This argument is only relevant when using the MG solver and should **only be set if the consequences to simulation and solver are evident**. A short description is given in Table 12.2.

Table 12.2: Preconditioner behaviour command summary

Value	Behaviour
STD	The preconditioner is rebuilt in every timestep (enabled by default)
HIERARCHY	The hierarchy (tentative prolongator) is reused
REUSE	The preconditioner is reused

12.14 Define the number of Energy Bins to use

Suppose dE the energy spread in the particle bunch is too large, the electrostatic approximation is no longer valid. One solution to that problem is to introduce k energy bins and perform k separate field solves in which dE is again small and hence the electrostatic approximation valid. In case of a cyclotron (Section 8.10) the number of energy bins must be at minimum the number of neighbouring bunches (NNEIGHBB) i.e. $ENBINS \leq NNEIGHBB$.

The variable MINSTEPFORREBIN defines the number of integration steps that have to pass until all energy bins are merged into one.

Chapter 13

Wakefields

Basically there are two different kind of wakefields that can be used. The first one is the wakefield of a round, metallic beam pipe that can be calculated numerically (see Sections 13.2 - 13.11). Since this also limits the applications of wakefields we also provide a way to import a discretized wakefield from a file (see Section 13.12).

The wakefield of a round, metallic beam pipe with radius a can be calculated by inverse FFT of the beam pipe impedance. There are known models for beam pipes with DC and AC conductivity. The DC conductivity of a metal is given by

$$\sigma_{DC} = \frac{ne^2\tau}{m} \quad (13.1)$$

with n the density of conduction electrons with charge e , τ the relaxation time, and m the electron mass. The AC conductivity, a response to applied oscillation fields, is given by

$$\sigma_{AC} = \frac{\sigma_{DC}}{1 - i\omega\tau} \quad (13.2)$$

with ω denoting the frequency of the fields.

The longitudinal impedance with DC conductivity is given by

$$Z_{Ldc}(k) = \frac{1}{ca} \frac{2}{\frac{\lambda}{k} - \frac{ika}{2}} \quad (13.3)$$

where

$$\lambda = \sqrt{\frac{2\pi\sigma|k|}{c}} (i + \text{sign}(k)) \quad (13.4)$$

with c denoting the speed of light and k the wave number.

The longitudinal wake can be obtained by an inverse Fourier transformation of the impedance. Since $\text{Re}(Z_L(k))$ drops at high frequencies faster than $\text{Im}(Z_L(k))$ the cosine transformation can be used to calculate the wake. The following equation holds in both, the DC and AC, case

$$W_L(s) = 10^{-12} \frac{2c}{\pi} \text{Re} \left(\int_0^\infty \text{Re}(Z_L(k)) \cos(ks) dk \right) \quad (13.5)$$

with $Z_L(k)$ either representing $Z_{LDC}(k)$ or $Z_{LAC}(k)$ depending on the conductivity. With help of the Panofsky-Wenzel theorem

$$Z_L(k) = \frac{k}{c} Z_T(k). \quad (13.6)$$

we can deduce the transversal wakefield from (13.5):

$$W_T(s) = 10^{-12} \frac{2c}{\pi} \text{Re} \left(\int_0^\infty \text{Re} \left(\frac{c}{k} Z_L(k) \right) \cos(ks) dk \right). \quad (13.7)$$

To calculate the integrals in (13.5) and (13.7) numerically the Simpson integration schema with equidistant mesh spacings is applied. This leads to an integration with small Δk with a big N which is computational not optimal with respect to efficiency. Since we calculate the wakefield usually just once in the initialization phase the overall performance will not be affected from this.

13.1 Wakefield Command

See Table 13.1 for a summary of the Wakefield command.

NOTE: Currently when using wakefields the domain can only be parallelized in z -direction! Please adapt the parallelization in the FieldSolver accordingly

```
Fs1:FIELDSOLVER, FSTYPE=FFT,
    MX=32, MY=32, MT=32,
    PARFFTX=false, PARFFTY=false, PARFFT=true,
    BCFFTX=open, BCFFTY=open, BCFFT=open,
    BBOXINCR=0, GREENSF=INTEGRATED;
```

Table 13.1: Wakefield command summary

Command	Purpose
WAKE	Specify a wakefield
TYPE	Specify the wake function [1D-CSR, LONG-SHORT-RANGE, TRANSV-SHORT-RANGE, LONG-TRANSV-SHORT-RANGE]
NBIN	Number of bins used in the calculation of the line density
CONST_LENGTH	TRUE if the length of the bunch is considered to be constant
CONDUCT	Conductivity [AC, DC]
Z0	Impedance of the beam pipe in [Ω]
FORM	The form of the beam pipe [ROUND]
RADIUS	The radius of the beam pipe in [m]
SIGMA	Material constant dependent on the beam pipe material in [$\Omega^{-1}m$]
TAU	Material constant dependent on the beam pipe material in [s]
FNAME	Specify a file that provides a wakefunction

13.2 Define the Wakefield to be used

The WAKE statement defines data for a wakefunction on an element.

13.3 Define the wakefield type

Used to specify the wakefunction

13.4 Define the number of bins

The number of bins used in the calculation of the line density.

13.5 Define the bunch length to be constant

With the `CONST_LENGTH` flag the bunch length can be set to be constant.

13.6 Define the conductivity

The conductivity of the bunch which can be set to either AC or DC.

13.7 Define the impedance

The impedance Z_0 of the beam pipe in $[\Omega]$.

13.8 Define the form of the beam pipe

The form of the beam pipe can be set to `ROUND`.

13.9 Define the radius of the beam pipe

The radius of the beam pipe in $[m]$.

13.10 Define the σ of the beam pipe

The σ of the beam pipe (material constant), see (13.1).

13.11 Define the relaxation time (τ) of the beam pipe

The τ defines the relaxation time and is needed to calculate the impedance of the beam pipe (see 13.1).

13.12 Import a wakefield from a file

Since we only need values of the wakefunction at several discrete points to calculate the force on the particle it is also possible to specify these in a file. To get required datapoints of the wakefield not provided in the file we linearly interpolate the available function values. The files are specified in the SDDS¹ (Self Describing Data Sets).

Whenever a file is specified OPAL will use the wakefield found in the file and ignore all other commands related to round beam pipes.

¹http://www.aps.anl.gov/Accelerator_Systems_Division/Operations_Analysis/manuals/GettingStartedWithSDDS/HTML/GettingStartedWithSDDS.html

13.13 Wake Functions

Three types of wake functions are implemented so far: transverse and longitudinal geometric wakes and the CSR wake. The general input format is

```
label:WAKE, TYPE=string, NBIN=real, CONST_LENGTH=bool,
      CONDUCT=string, Z0=real, FORM=string, RADIUS=real,
      SIGMA=real, TAU=real, FILTERS=string-array
```

CONST_LENGTH, **CONDUCT**, **Z0**, **FORM**, **RADIUS**, **SIGMA** and **TAU** are only used in the geometric wakes.

TYPE The type of wake function; either TRANSV-SHORT-RANGE, LONG-SHORT-RANGE or 1D-CSR.

NBIN Not implemented yet; Number of bins to be used for line density if different from space charge solver grid.

CONST_LENGTH

CONDUCT

Z0

FORM

RADIUS

SIGMA

TAU

FILTERS Array of names of filters to be applied onto the longitudinal histogram of the bunch to get rid of the noise and to calculate derivatives. All the filters/smoothers are applied to the line density in the order they appear in the array. The last filter is also used for calculating the derivatives. The actual filters have to be defined elsewhere.

13.14 Filters

Filters can be defined which then are applied to the line density of the bunch. The following smoothing filters are implemented: Savitzky-Golay, Stencil, FixedFFTLowPass, RelativFFTLowPass. The input format for them is

```
label:FILTER, TYPE=string, NFREQ=real, THRESHOLD=real,
      NPOINTS=real, NLEFT=real, NRIGHT=real,
      POLYORDER=real
```

TYPE The type of filter: Savitzky-Golay, Stencil, FixedFFTLowPass, RelativFFTLowPass

NFREQ Only used in FixedFFTLowPass: the number of frequencies to keep

THRESHOLD Only used in RelativeFFTLowPass: the minimal strength of frequency compared to the strongest to consider.

NPOINTS Only used in Savitzky-Golay: width of moving window in number of points

NLEFT Only used in Savitzky-Golay: number of points to the left

NRIGHT Only used in Savitzky-Golay: number of points to the right

POLYORDER Only used in Savitzky-Golay: polynomial order to be used in least square approximation

The Savitzky-Golay filter and the ones based on the FFT routine provide a derivative on a natural way. For the Stencil filter a second order stencil is used to calculate the derivative.

An implementation of the Savitzky-Golay filter can be found in the Numerical Recipes. The Stencil filter uses the following two stencil consecutively to smooth the line density:

$$f_i = \frac{7 \cdot f_{i-4} + 24 \cdot f_{i-2} + 34 \cdot f_i + 24 \cdot f_{i+2} + 7 \cdot f_{i+4}}{96}$$

and

$$f_i = \frac{7 \cdot f_{i-2} + 24 \cdot f_{i-1} + 34 \cdot f_i + 24 \cdot f_{i+1} + 7 \cdot f_{i+2}}{96}.$$

For the derivative a standard second order stencil is used:

$$f'_i = \frac{f_{i-2} - 8 \cdot f_{i-1} + 8 \cdot f_{i+1} - f_{i+2}}{h}$$

This filter was designed by Ilya Pogorelov for the ImpactT implementation of the CSR 1D model.

The FFT based smoothers calculate the Fourier coefficients of the line density. Then they set all coefficients corresponding to frequencies above a certain threshold to zero. Finally the back-transformation is calculate using this coefficients. The two filters differ in the way they identify coefficients which should be set to zero. FixedFFTLowPass uses the n lowest frequencies whereas RelativeFFTLowPass searches for the coefficient which has the biggest absolut value. All coefficients which, compared to this value, are below a threshold (measure in percents) are set to zero. For the derivative the coefficients are multiplied with the following function (this is equivalent to a convolution):

$$g_i = \begin{cases} i \frac{2\pi i}{N \cdot L} & i < N/2 \\ -i \frac{2\pi i}{N \cdot L} & i > N/2 \end{cases}$$

where N is the total number of coefficients/sampling points and L is the length of the bunch.

Chapter 14

Geometry

At present the GEOMETRY command is still an **experimental feature** which is not to be used by the general user. It can only be used to specify boundaries for the MG Solver. The command can be used in two modes:

1. specify a H5FED file holding the surface mesh of a complicated boundary geometry
2. specify a cylinder with an elliptic base area

14.1 Geometry Command

Table 14.1: Geometry command summary

Command	Purpose
GEOMETRY Specify a geometry	
FGEOM	Specifies the H5FED geometry file
LENGTH	Specifies the length of the geometry
S	Specifies the start of the geometry
A	Specifies the semi-major axis of the elliptic base area
B	Specifies the semi-minor axis of the elliptic base area

14.2 Define the Geometry File

The H5FED file containing the surface mesh of the geometry.

14.3 Define the Length

The length of the specified geometry in [m].

14.4 Define the Start

The start of the specified geometry in [m].

14.5 Define the Semi-Major Axis

The semi-major axis of the ellipse in [m].

14.6 Define the Semi-Minor Axis

The semi-minor axis of the ellipse in [m].

Chapter 15

Tracking

Table 15.1: Commands accepted in Tracking Mode

Command	Purpose
TRACK	Enter tracking mode
LINE	Label of LINE or SEQUENCE
BEAM	Label of BEAM
T0	Initial time
DT	Array of time step sizes for tracking
MAXSTEPS	Array of maximal number of time steps
ZSTOP	Array of z-location [m], after which the simulation switches to the next set of DT, MAXSTEPS and ZSTOP
STEPSPERTURN	The timesteps per revolution period
TIMEINTEGRATOR	Defines the time integrator used in OPAL-CYCL
name=expression	Parameter relation
ENDTRACK	Leave tracking mode

15.1 Track Mode

Before starting to track, a beam line (see §9) and a beam (see §10.1) must be selected. The time step (DT) and the maximal steps to track (MAXSTEPS) or ZSTOP should be set. This command causes OPAL to enter “tracking mode”, in which it accepts only the track commands (see Tab. 15.1). Several tracks can be defined in a sequence and all parameters are always local to the actual step.

The attributes of the command are:

LINE The label of a preceding LINE (see §9) (no default).

BEAM The named BEAM command defines the particle mass, charge and reference momentum (default: UNNAMED_BEAM).

T0 The initial time [s] of the simulation, its default value is 0.

DT Array of time step sizes for tracking, default length of the array is 1 and its only value is 1 ps.

MAXSTEPS Array of maximal number of timesteps, default length of the array is 1 and its only value is 10.

ZSTOP Array of z-locations [m], default length of the array is 1 and its only value is $1E6$ [m]. The simulation switches to the next set, $i + 1$, of DT, MAXSTEPS and ZSTOP if either it has been tracking with the current set for more than $MAXSTEPS_i$ steps or the mean position has reached a z-position larger than $ZSTOP_i$. If set i is the last set of the array then the simulation stops.

TIMEINTEGRATOR Define the time integrator. Currently only available in OPAL-CYCL. The valid options are RK-4, LF-2 and MTS:

- RK-4: the fourth-order Runge-Kutta integrator. This is the default integrator for OPAL-CYCL.
- LF-2: the second-order Boris-Buneman (leapfrog-like) integrator. Currently, LF-2 is only available for multi-particles with/without space charge. For single particle tracking and tune calculations, use the RK-4 for the time being.
- MTS: the multiple-time-stepping integrator. Considering that the space charge fields change much slower than the external fields in cyclotrons, the space charge can be calculated less frequently than the external field interpolation, so as to reduce time to solution. The outer step (determined by STEPSPERTURN) is used to integrate space charge effects. A constant number of substeps per outer step is used to query external fields and to move the particles. The number of substeps can be set with the option MTSSUBSTEPS and its default value is 1. When using this integrator, the input file has to be rewritten in the units of the outer step. For example, extracts of the inputfile suited for LF-2 or RK-4 read

```
Option, PSDUMPFREQ=100;
Option, REPARTFREQ=20;
Option, SPTDUMPFREQ=50;
turns=5;
nstep=3000;
TRACK, LINE=11, BEAM=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep,
TIMEINTEGRATOR="LF-2";
      RUN, METHOD = "CYCLOTRON-T", BEAM=beam1, FIELDSOLVER=Fsl, DISTRIBUTION=Dist1;
ENDTRACK;
```

and should be transformed to

```
Option, MTSSUBSTEPS=10;
Option, PSDUMPFREQ=10;
Option, REPARTFREQ=2;
Option, SPTDUMPFREQ=5;
turns=5;
nstep=300;
TRACK, LINE=11, BEAM=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep,
TIMEINTEGRATOR="MTS";
      RUN, METHOD = "CYCLOTRON-T", BEAM=beam1, FIELDSOLVER=Fsl, DISTRIBUTION=Dist1;
ENDTRACK;
```

In general all step quantities should be divided by MTSSUBSTEPS.

In our first experiments on PSI injector II cyclotron, simulations with reduced space charge solving frequency by a factor of 10 lie still very close to the original solution. How large MTSSUBSTEPS can be chosen of course depends on the importance of space charge effects.

STEPSPERTURN The timesteps per revolution period. Only available for OPAL-CYCL, default value is 720.

In OPAL-T and OPAL-MAP, the command format is:

```
TRACK, LINE=name, BEAM=name, MAXSTEPS=value, DT=value;
```

In OPAL-CYCL, instead of setting time step, the timesteps per-turn should be set. The command format is:

```
TRACK, LINE=name, BEAM=name, MAXSTEPS=value, STEPSPERTURN=value;
```

Particles are tracked in parallel i.e. the coordinates of all particles are transformed at each beam element as it is reached.

OPAL leaves **track mode** when it sees the command

```
ENDTRACK;
```

15.1.1 Track a Random Machine

This example shows how to track a *random* machine i.e. some parameters are random variables. At the moment (Version 1.1.4) there seems to be a problem when having random variables in the Distribution command.

```
Option, SCAN=TRUE;
```

```
.....
```

```
I=0;
```

```
WHILE (I < 3) {
```

```
    rv1:= (RANF() * 4.7);
```

```
    rv2:=0.0;
```

```
    rv3:=0.0;
```

```
    rv4:=0.0;
```

```
    rv5:=0.0;
```

```
    Ppo: PepperPot, L=200.0E-6, ELEMEDGE=6.0E-3,
        R=1.0E-4, PITCH=0.5E-4, NHOLX=20, NHOLY=20,
        XSIZE=5.0E-3, YSIZE=5.0E-3, OUTFN="ppo.h5";
```

```
    Col: ECollimator, L=3.0E-3, ELEMEDGE=7.0E-3,
        XSIZE=7.5E-4, YSIZE=7.5E-4, OUTFN="Coll.h5";
```

```
    SP1: Solenoid, L=1.20, ELEMEDGE=-0.5315,
        FMAPFN="1T2.T7", KS=8.246e-05 + rv2;
```

```
    SP2: Solenoid, L=1.20, ELEMEDGE=-0.397,
        FMAPFN="1T3.T7", KS=1.615e-05 + rv3;
```

```
    SP3: Solenoid, L=1.20, ELEMEDGE=-0.267,
        FMAPFN="1T3.T7", KS=1.016e-05 + rv4;
```

```
    SP4: Solenoid, L=1.20, ELEMEDGE=-0.157,
        FMAPFN="1T3.T7", KS=4.750e-05 + rv5;
```

```
    SP5: Solenoid, L=1.20, ELEMEDGE=-0.047,
        FMAPFN="1T3.T7", KS=0.0;
```

```
    gun: RFCavity, L=0.013, VOLT=(-47.51437343 + rv1),
        FMAPFN="1T1.T7", ELEMEDGE=0.00,
        TYPE="STANDING", FREQ=1.0e-6;
```

```
    value,{I, rv1, rv2, rv3, rv4, rv5};
```

```
    ll: Line=(gun, Ppo, sp1, sp2, sp3, sp4, sp5);
```

```
SELECT, Line=l1;
TRACK, line=l1, beam=beam1, MAXSTEPS=500, DT=2.0e-13;
  RUN, method="PARALLEL-T", beam=beam1,
    fieldsolver=Fsl, distribution:=Dist1;
ENDTRACK;

SYSTEM,"mkdir -p scan0-" & STRING(I);
SYSTEM,"mv scan-0.h5 scan-0.stat scan-0.lbal scan0-"
      & STRING(I);
I=EVAL(I+1.0);
}
```

Chapter 16

Field Emission

Field emission is a major source of both dark current particles and primary incident particles in secondary emission. The Fowler-Nordheim (F-N) formula we use here to predict the emitted current density is given in (16.1) [53] [54]

$$J(\mathbf{r}, t) = \frac{A(\beta E)^2}{\varphi t(y)^2} \exp\left(\frac{-Bv(y)\varphi^{3/2}}{\beta E}\right) [\text{A/m}^2] \quad (16.1)$$

where $J(\mathbf{r}, t)$ stands for emitted electric current density in position \mathbf{r} and time t . The Greek letters φ and β denote the work function of the surface material and the local field enhancement factor respectively. The parameter E is the electric field in the normal direction of surface. The parameters A and B are empirical constants. The functions $v(y)$ and $t(y)$ representing the image charge effects [53] as a function of the Fowler-Nordheim parameter y with the following definition[55]

$$y = \sqrt{\frac{e^3}{4\pi\epsilon}} \frac{\sqrt{\beta E}}{\varphi} = 3.795 \times 10^{-5} \frac{\sqrt{\beta E}}{\varphi}. \quad (16.2)$$

In our model, we have chosen a simpler approximation originated by J. H. Han[55]

$$\begin{aligned} v(y) &= a - by^2 \\ t(y) &\approx 1. \end{aligned}$$

These approximations are valid for a large range of y , corresponding to typical applied electric field ranges in RF guns.

Whenever the normal components of an electric field are strong enough the field emission current density will be limited by space charge effect[53]. To cover this situation we incorporated the 1D Child-Langmuir law

$$\begin{aligned} J(\mathbf{r}, t) &= \frac{4\epsilon_0}{9} \sqrt{2\frac{e}{m}} \left(\frac{V^{3/2}}{d^2} \right) \\ &= \frac{4\epsilon_0}{9} \sqrt{2\frac{e}{m}} \left(\frac{E^{3/2}}{d^{1/2}} \right) [\text{A/m}^2] \end{aligned} \quad (16.3)$$

into our field emission model. $J(\mathbf{r}, t)$ denotes space charge limited emission current density in position \mathbf{r} and time t , ϵ_0 the permittivity in vacuum, E the normal component of electric field on the surface and d the distance from the position where E is evaluated. Currently we choose d to be equal to the distance traveled by emitted particles in one time step, i.e., $d = \frac{eE\Delta t^2}{2m_0}$ where Δt is simulation time step.

16.1 Field Emission Command

To perform field emission related simulation, a triangulated surface geometry defined by `GEOMETRY` command (see Chapter 14) should be specified and attached to the elements (currently only `RFCavity` element is valid for field emission). A `SURFACEEMISSION` type of distribution, defined in `DISTRIBUTION` command should be attached to the `GEOMETRY` command. And users can customize dark current simulation by specifying the value of the work function φ , local field enhancement factor β and other parameters present in (16.1) and (16.2) in the `SURFACEEMISSION` type of distribution definition in input file. See the following example input file and Table 16.1 for a summary of the field emission related command in the `SURFACEEMISSION` type of distribution definition.

```
DistSurf: DISTRIBUTION, DISTRIBUTION = "SURFACEEMISSION",
          NPDARKCUR = 0, INWARDMARGIN=0.0,
          FNBETA = 30, FNMAXEMI = 2,
          FNFIELDTHR = -0.1;
ge:      GEOMETRY, FGEOM="../New_Gun.h5",
          S=0.0, DISTR=DistSurf,
          ZSHIFT=0.0;
FINSSGUN: RFCavity, L = 0.175,
          VOLT = 100.0, FMAPFN = "../RF_GUN_PSI-fieldmap.T7" ,
          GEOMETRY = ge, ELEMEDGE =0.0,
          TYPE = "STANDING", FREQ = 2997.922938148;

\ldots
```

Table 16.1: Field Emission Command summary

Command	Purpose (Default)
FNA	Empirical constant A for F-N emission model (1.54×10^{-6})
FNB	Empirical constant B for F-N emission model (6.83×10^9)
FNY	Constant for image charge effect parameter $y(E)$ (3.795×10^{-5})
FNZYZERO	Zero order constant for $v(y)$ function (0.9632)
FNZYSECOND	Second order constant for $v(y)$ function (1.065)
FNPHIW	Work function of gun surface material (4.65 eV)
FNBETA	Field enhancement factor β for F-N emission (50.0)
FNFIELDTHR	Field threshold for F-N emission (30.0 MV/m)
FNMAXEMI	Maximum Number of electrons emitted from a single triangle in each time step (10)

Chapter 17

Multipacting

Multiple electron impacting (multipacting) is a phenomenon in radio frequency (RF) structure that under certain conditions (material and geometry of the RF structure, frequency and level of the electromagnetic field, with or without the appearance of the magnetic field . . .), electrons secondary emission yield (SEY) coefficient will be larger than one and lead to exponential multiplication of electrons.

Besides the particle tracker in OPAL, the computational model for solving multipacting problem contains an accurate representation of 3D geometry of RF structure by using triangulated surface mesh (see Chapter 14 and Chapter 16), an efficient particle-boundary collision test scheme, two different secondary emission models, and necessary post-processing scripts.

As we use a triangulated surface mesh to represent the RF structure, our particle-boundary collision test scheme is based on line segment-triangle intersection test. An axis aligned boundary box combined with surface triangle inward normal method is adopted to speedup the particle-boundary collision test [56].

The SEY curve is a very important property of the surface material for the development of a multipacting in a RF structure. Figure 17.1 shows a typical SEY curve. Here, the horizontal axis is the energy of impacting electron, the vertical axis is the SEY value δ , defined as [57]:

$$\delta = \frac{I_s}{I_0} \quad (17.1)$$

where I_0 is the incident electron beam current and I_s is the secondary current, i.e., the electron current emitted from the surface. Usually the SEY value δ appeared in an SEY curve is the measured SEY with normal incident, i.e., the impacting electron is perpendicular to the surface. The energy E_1 and E_2 are the first crossover energy and the second crossover energy respectively, where the SEY value δ exceed and fall down to $\delta = 1$ at the first time. Obviously, only the energy range of $\delta > 1$, i.e., $E \in (E_1, E_2)$ can contribute to multipacting.

Both Furman-Pivi's probabilistic secondary emission model [57] and Vaughan's formula based secondary emission model [58] have been implemented in OPAL and have been benchmarked (see Section 17.2).

The Furman and Pivi's secondary emission model calculates the number of secondary electrons that result from an incident electron of a given energy on a material at a given angle (see Figure 17.2). For each of the generated secondary electrons the associated process: *true secondary*, *rediffused* or *backscattered* is recorded, as is sketched in Figure 17.2. This model is mathematically self-consistent, which means that (1) when averaging over an infinite number of secondary-emission events, the reconstructed δ and $d\delta/dE$ are guaranteed to agree with the corresponding input quantities; (2) the energy integral of $d\delta/dE$ is guaranteed to equal δ ; (3) the energy of any given emitted electron is guaranteed not to exceed the primary energy; and (4) the aggregate energy of the electrons emitted in any multielectron event is also guaranteed not to exceed the primary energy. This model contains built-in SEY curves for copper and stainless steel and the only thing user need to set is to choose the material type, i.e., copper or stainless steel, as long as the surface material of user's RF structure has the same SEY curve as built-in SEY curves.

Although a set of parameters in the model can be adjusted to model different SEY curves without breaking the

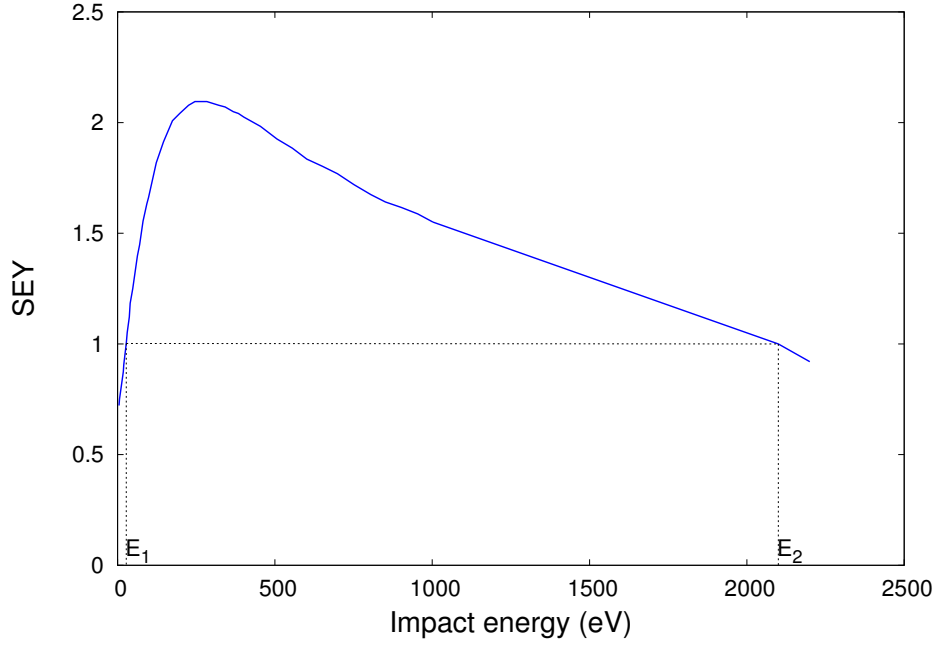


Figure 17.1: Typical SEY curve

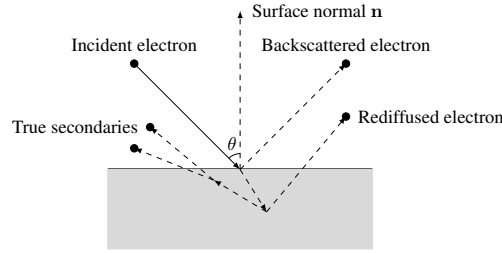


Figure 17.2: Sketch map of the secondary emission process.

above mentioned mathematical self-consistency, it is easier to use Vaughan's formula based secondary emission model if user has to model a different SEY curve.

The Vaughan's secondary emission model is based on a secondary emission yield formula [58, 59]:

$$\delta(E, \theta) = \delta_{max}(\theta) \cdot (ve^{1-v})^k, \text{ for } v \leq 3.6 \quad (17.2a)$$

$$\delta(E, \theta) = \delta_{max}(\theta) \cdot 1.125/v^{0.35}, \text{ for } v > 3.6 \quad (17.2b)$$

$$\delta(E, \theta) = \delta_0, \text{ for } v \leq 0 \quad (17.2c)$$

where

$$v = \frac{E - E_0}{E_{max}(\theta) - E_0},$$

$$k = 0.56, \text{ for } v < 1,$$

$$k = 0.25, \text{ for } 1 < v \leq 3.6,$$

$$\delta_{max}(\theta) = \delta_{max}(0) \cdot (1 + k_\theta \theta^2 / 2\pi),$$

$$E_{max}(\theta) = E_{max}(0) \cdot (1 + k_E \theta^2 / 2\pi).$$

The secondary emission yield value for an impacting electron energy E and incident angle θ w.r.t the surface normal is denoted as $\delta(E, \theta)$. Parameter k_θ and k_E denotes the dependence on surface roughness. Both should be assigned a default value of 1.0, which appears appropriate for typical dull surfaces in a working tube environment, and lower values down to zero or higher values, up to about 2.0, are only valid for specified cases [58]. $E_{max}(0)$ is the impacting energy when incident angle is zero and secondary yield reaches its maximum value. E_0 is an adjustable parameter to make the first crossover energy be fitted to the experiment data [60]. δ_0 is the user specified constant denote the SEY of low impacting energy.

The emission energy obeys the thermal energy distribution:

$$f(\varepsilon) = \frac{\varepsilon}{(k_B T)^2} \exp\left(\frac{-\varepsilon}{k_B T}\right) \quad (17.3)$$

The polar angles of emitted secondaries $\theta \in [0, \pi/2]$ are with probability density $\cos^\alpha \theta$, and the azimuthal angles $\phi \in [0, 2\pi]$ are with uniform probability density. These emission angles of the secondary electrons is relative to the local coordinate system which is centered at the collision point and whose z axis is along the inward normal of the surface.

Motivated by the fact that the population of particles in simulation domain may continually grow exponentially and lead to tens of magnitude larger within limited simulation time steps, which may cause the exhaust of computing memory, a re-normalization of simulation particle number approach is also implemented. In each electron impacting events, instead of emitting the real number of simulation particles predicted by secondary emission models, this re-normalization approach emit only one particle, and the current of newly emitted particle will be the current of incident particle multiplied by SEY value δ .

17.1 Commands Related to Multipacting Simulation

To perform multipacting simulation, a triangulated surface geometry defined in the `GEOMETRY` command (see Chapter 14) must be specified and attached to the elements (currently only `RFCavity`, `ParallelPlate` and `Drift` elements are available for multipacting simulation).

A distribution array, containing `SURFACEEMISSION` and `SURFACERANDCREATE` type of distributions, defined in the `DISTRIBUTION` command must be attached to the `GEOMETRY`. Users can use commands contained in `SURFACERANDCREATE` type of distribution to specify the position of initial *seed* electrons. And commands within `SURFACEEMISSION` type of distribution can be used to customize the type and the parameters of secondary emission model in input file.

A summary of multipacting simulation related parameters are given in Table 17.1.

The following example shows the usage of the multipacting simulation related command.

```
Title, string="Cyclotron_Multipacting_Simulation_example";

Option, ECHO=FALSE;
Option, INFO=FALSE;

Option, PSDUMPFREQ=1;
Option, STATDUMPFREQ=1;
Option, PPDEBUG=FALSE;
Option, SURFDUMPFREQ=100;

// Set an upper limit of simulation particle number
// to prevent memory overflow.

MAXPARTSNUM=1000000;

// SECONDARYFLAG = 1: Using Furman-Pivi's model
// SURFMATERIAL=0: surface material is copper
// Set NEMISSIONMODE=false will use re-normalize
// simulation particle approach.
// Set the field enhancement factor FNBETA to
// a very small number to prevent field
// emission.

DistSurf: DISTRIBUTION, DISTRIBUTION = "SURFACEEMISSION",
      NPDARKCUR =0, INWARDMARGIN = 0.0,
      FNBETA = 0.1, FNMAXEMI = 2,
      SECONDARYFLAG = 1,
      NEMISSIONMODE=false,
      SURFMATERIAL=0;

// INWARDMARGIN: seed electron positions along the
// inward normal w.r.t the boundary surface.
DistSurf1: DISTRIBUTION,
      DISTRIBUTION = "SURFACERANDCREATE",
      INWARDMARGIN = 0.0, NPDARKCUR =10000,
      EINITHR = 0.2;

// For multipacting study of cyclotron cavity,
// the axis z in geometry file is actually axis y
// in ParallelTracker, so we need to shift z
// coordinates of the geometry by specifying ZSHIFT to make
// sure that the z coordinates read in by ParallelTracker
// will be correct.

ge: GEOMETRY, FGEOM="../../10.h5", S=0.0,
      ZSHIFT=0.631, DISTRs={DistSurf, DistSurf1};
```

```

Box: RFCavity, PLENGTH = 1.262, VOLT = 1,
      GEOMETRY = ge, FMAPFN = "../CyciaeEM.h5",
      ELEMEDGE = 0, FAST=true,
      FREQ = 44.6, LAG = 0.0,
      DX = 0, DY = 0;

// This element is used to model the magnetic field in
// the valley of a cyclotron, where the RF cavity is installed.

Mag: CYCLOTRONVALLEY, FMAPFN = "../CyciaeMagReal.h5",
      ELEMEDGE = 0, DX = 0, DY = 0;

Benchmark: Line = (Box, Mag);

Fs1:FIELDSOLVER, FSTYPE = NONE, MX = 32,
      MY = 32, MT = 256,
      PAREFTX = true, PARFFTY = true,
      PARFFTT = false, BCFFTX = open,
      BCFFTY = open, BCFFTT = open,
      BBOXINCR = 0.1, GREENSF = INTEGRATED;

qb=0.2e-9;
bfreq=300;
bcurrent=qb*bfreq;

beam1: BEAM, PARTICLE = ELECTRON, pc = P0,
      NPART = 2000, BFREQ = bfreq ,
      BCURRENT = bcurrent, CHARGE = -1;

Select, Line=Benchmark;

track, line= Benchmark, beam=beam1,
      MAXSTEPS=23000, DT=4e-12, ZSTOP=3;

run, method = "PARALLEL-T", beam = beam1,
      fieldsolver = Fs1, MULTIPACTING=true;
endtrack;

Quit;

```

Table 17.1: Multipacting Related Command Summary

Command	Purpose (Default)
VW	Velocity scalar in Maxwellian Dist (1.0 m/s)
VVTHERMAL	Thermal velocity in Maxwellian Dist (7.268929821×10^5 m/s)
SECONDARYFLAG	Secondary model type, 0:none, 1:Furman-Pivi, 2:Vaughan (0)
NEMISSIONMODE	Emit real No. secondaries or not (true)
VEZERO	SEY will be δ_0 , if energy is less than VEZERO in Vaughan's model (12.5 eV)
VSEYZERO	δ_0 in Vaughan's model (0.5)
VSEYMAX	δ_{max} in Vaughan's model (2.22)
VEMAX	Energy related to δ_{max} in Vaughan's model (165 eV)
VKENERGY	The roughness of surface for impact energy in Vaughan's model(1.0)
VKTHETA	The roughness of surface for impact angle in Vaughan's model(1.0)
SURFMATERIAL	The material type for Furman-Pivi model, 0: copper; 1: stainless steel (0)

17.2 Run Parallel Plate Benchmark

Both the Furman-Pivi's model and Vaughan's model have been carefully benchmarked in both re-normalize simulation particle approach and real simulation particles approach against a non-stationary multipacting theory [61]. The OPAL simulation results and the theory match very well (see figure 17.3 and figure 17.4).

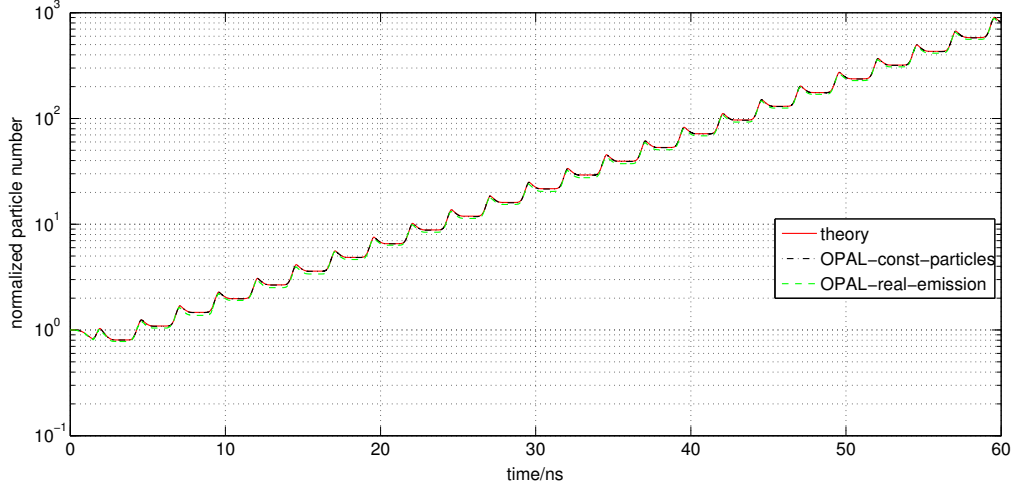


Figure 17.3: Time evolution of electron number predicted by theoretical model and OPAL simulation using Furman-Pivi's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 200\text{MHz}$, $V_0 = 120\text{V}$, $d = 5\text{mm}$

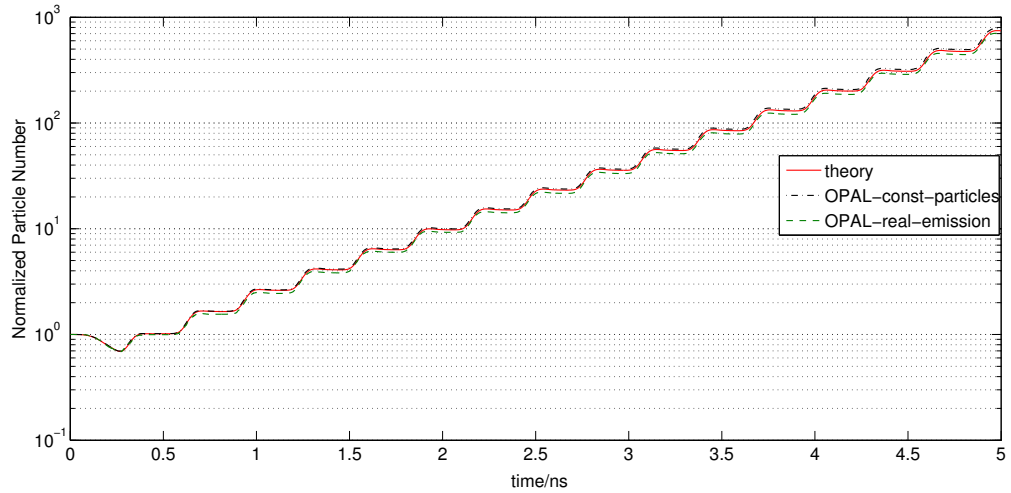


Figure 17.4: Time evolution of electron number predicted by theoretical model and OPAL simulation using Vaughan's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 1640\text{MHz}$, $V_0 = 120\text{V}$, $d = 1\text{mm}$.

To run the parallel plate benchmark simulation, user need to set the option `PPDEBUG` in the input file `true`. The

input file and the geometry file needed by the parallel plate benchmark simulation can be found in the regression test folder.

17.3 PostProcessing

In the general case (not only in multipacting simulations), OPAL will dump the 6D phase space and statistical information of the particles in the simulation domain, into a `h5` file. The dump frequency, i.e., after how many time steps the particle information will be saved can be specified with the option `PSDUMPFREQ`. Setting `Option, PSDUMPFREQ=1` dumps the information in each time step.

A utility tool `h5ToVtk` converts the `h5` file to the Visualization Toolkit (VTK) legacy format. The number of VTK files equals to the number of time steps in `h5` file. These VTK files together with a VTK file automatically generated by the geometry class of OPAL which contains the geometry of the RF structure under study can be visualized using for example with Paraview [62]. The animation and clip feature of Paraview is very useful to visualize the particle motion inside the RF structure.

For simulations involving the geometry (multipacting and field emission), OPAL will also dump the position and current of incident particles into another `h5` file with the name `*_Surface.h5`, where the asterisk stands for the base name of the user's OPAL input file. If we need this surface loss data during post processing, we should specify the dump frequency in the option `SURFDUMPFREQ` with a positive integer in the OPAL input file, otherwise, the default value of the option is `SURFDUMPFREQ=-1`, and the `*_Surface.h5` will not be generated. Another utility tool `h5SurfaceVtk` convert the `*_Surface.h5` file to VTK files. For multipacting simulation, these VTK files can be used to visualize the *hot spots* of the RF structure where multipacting happens.

The above mentioned utility tools are based on H5hut library, and will soon be available in the distribution.

Some of the boundary geometry related simulations, like the multipacting simulation using re-normalizing particle number approach, or dark current simulations where the current of field emitted particles from a single triangle has been re-normalized as the model predicted current has exceeded the user defined upper limit, the current (weight) of simulation particles varies and each simulation particle stands for more physical particles than the initial simulation particles. In these cases, instead of using simulation particles, we count the number of *effective particles* defined as the ratio of total current in simulation over the current of a single initial particle.

An ASCII file named `Part_statistics.dat` containing the simulation time, the number of impacts and associated total SEY value as well as the number of *effective particles* in each time step. This makes the analysis of the time evolution of particle density feasible with tools like GNUPLOT.

Chapter 18

Physics Models Used in the Particle Matter Interaction Model

18.1 The Energy Loss

The energy loss is simulated using the Bethe-Bloch equation.

$$-dE/dx = \frac{Kz^2Z}{A\beta^2} \left[\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 \right], \quad (18.1)$$

where Z is the atomic number of absorber, A is the atomic mass of absorber, m_e is the electron mass, z is the charge number of the incident particle, $K = 4\pi N_A r_e^2 m_e c^2$, r_e is the classical electron radius, N_A is the Avogadro's number, I is the mean excitation energy. β and γ are kinematic variables. T_{max} is the maximum kinetic energy which can be imparted to a free electron in a single collision.

$$T_{max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma m_e/M + (m_e/M)^2}, \quad (18.2)$$

where M is the incident particle mass.

The stopping power is compared with PSTAR program of NIST in Fig. 18.1.

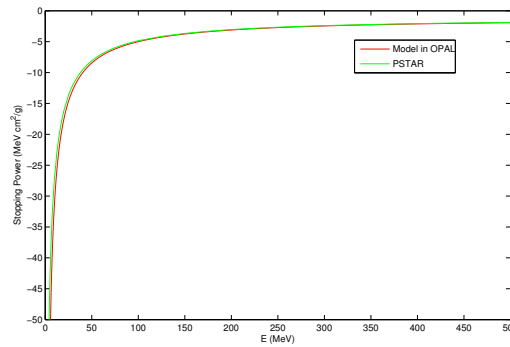


Figure 18.1: The comparison of stopping power with PSTAR.

Energy straggling: For relatively thick absorbers such that the number of collisions is large, the energy loss distribution is shown to be Gaussian in form. For nonrelativistic heavy particles the spread σ_0 of the Gaussian

distribution is calculated by:

$$\sigma_0^2 = 4\pi N_A r_e^2 (m_e c^2)^2 \rho \frac{Z}{A} \Delta s, \quad (18.3)$$

where ρ is the density, Δs is the thickness.

18.2 The Coulomb Scattering

The Coulomb scattering is treated as two independent events: the multiple Coulomb scattering and the large angle Rutherford scattering.

Using the distribution given in Classical Electrodynamics, by J. D. Jackson, the multiple- and single-scattering distributions can be written:

$$P_M(\alpha) d\alpha = \frac{1}{\sqrt{\pi}} e^{-\alpha^2} d\alpha, \quad (18.4)$$

$$P_S(\alpha) d\alpha = \frac{1}{8 \ln(204 Z^{-1/3})} \frac{d\alpha}{\alpha^3}, \quad (18.5)$$

where $\alpha = \frac{\theta}{\langle \Theta^2 \rangle^{1/2}} = \frac{\theta}{\sqrt{2}\theta_0}$.

the transition point is $\theta = 2.5\sqrt{2}\theta_0 \approx 3.5\theta_0$,

$$\theta_0 = \frac{13.6 \text{ MeV}}{\beta c p} z \sqrt{\Delta s / X_0} [1 + 0.038 \ln(\Delta s / X_0)], \quad (18.6)$$

where p is the momentum, Δs is the stepsize, and X_0 is the radiation length.

18.2.1 Multiple Coulomb Scattering

Generate two independent Gaussian random variables with mean zero and variance one: z_1 and z_2 . If $z_2\theta_0 > 3.5\theta_0$, start over. Otherwise,

$$x = x + \Delta s p_x + z_1 \Delta s \theta_0 / \sqrt{12} + z_2 \Delta s \theta_0 / 2, \quad (18.7)$$

$$p_x = p_x + z_2 \theta_0. \quad (18.8)$$

Generate two independent Gaussian random variables with mean zero and variance one: z_3 and z_4 . If $z_4\theta_0 > 3.5\theta_0$, start over. Otherwise,

$$y = y + \Delta s p_y + z_3 \Delta s \theta_0 / \sqrt{12} + z_4 \Delta s \theta_0 / 2, \quad (18.9)$$

$$p_y = p_y + z_4 \theta_0. \quad (18.10)$$

18.2.2 Large Angle Rutherford Scattering

Generate a random number ξ_1 , if $\xi_1 < \frac{\int_{2.5}^{\infty} P_S(\alpha) d\alpha}{\int_0^{2.5} P_M(\alpha) d\alpha + \int_{2.5}^{\infty} P_S(\alpha) d\alpha} = 0.0047$, sampling the large angle Rutherford scattering.

The cumulative distribution function of the large angle Rutherford scattering is

$$F(\alpha) = \frac{\int_{2.5}^{\alpha} P_S(\alpha) d\alpha}{\int_{2.5}^{\infty} P_S(\alpha) d\alpha} = \xi, \quad (18.11)$$

where ξ is a random variable. So

$$\alpha = \pm 2.5 \sqrt{\frac{1}{1 - \xi}} = \pm 2.5 \sqrt{\frac{1}{\xi}}. \quad (18.12)$$

Generate a random variable P_3 ,
 if $P_3 > 0.5$

$$\theta_{Ru} = 2.5 \sqrt{\frac{1}{\xi}} \sqrt{2} \theta_0, \quad (18.13)$$

else

$$\theta_{Ru} = -2.5 \sqrt{\frac{1}{\xi}} \sqrt{2} \theta_0. \quad (18.14)$$

The angle distribution after Coulomb scattering is shown in Fig. 18.2. The line is from Jackson's formula, and the points are simulations with Matlab. For a thickness of $\Delta s = 1e-4$ m, $\theta = 0.5349\alpha$ (in degree).

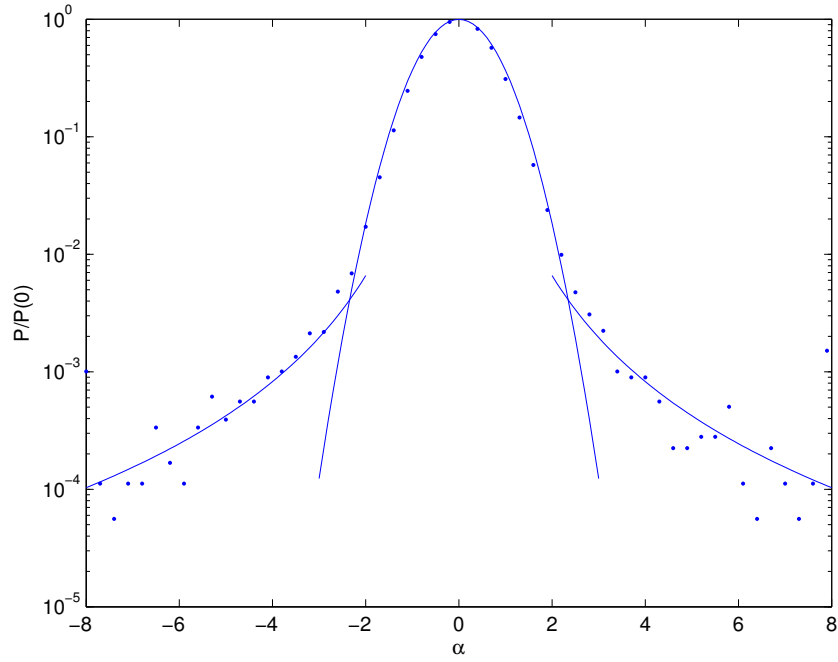


Figure 18.2: The comparison of Coulomb scattering with Jackson's book.

18.3 The Flow Diagram of *CollimatorPhysics* Class in OPAL

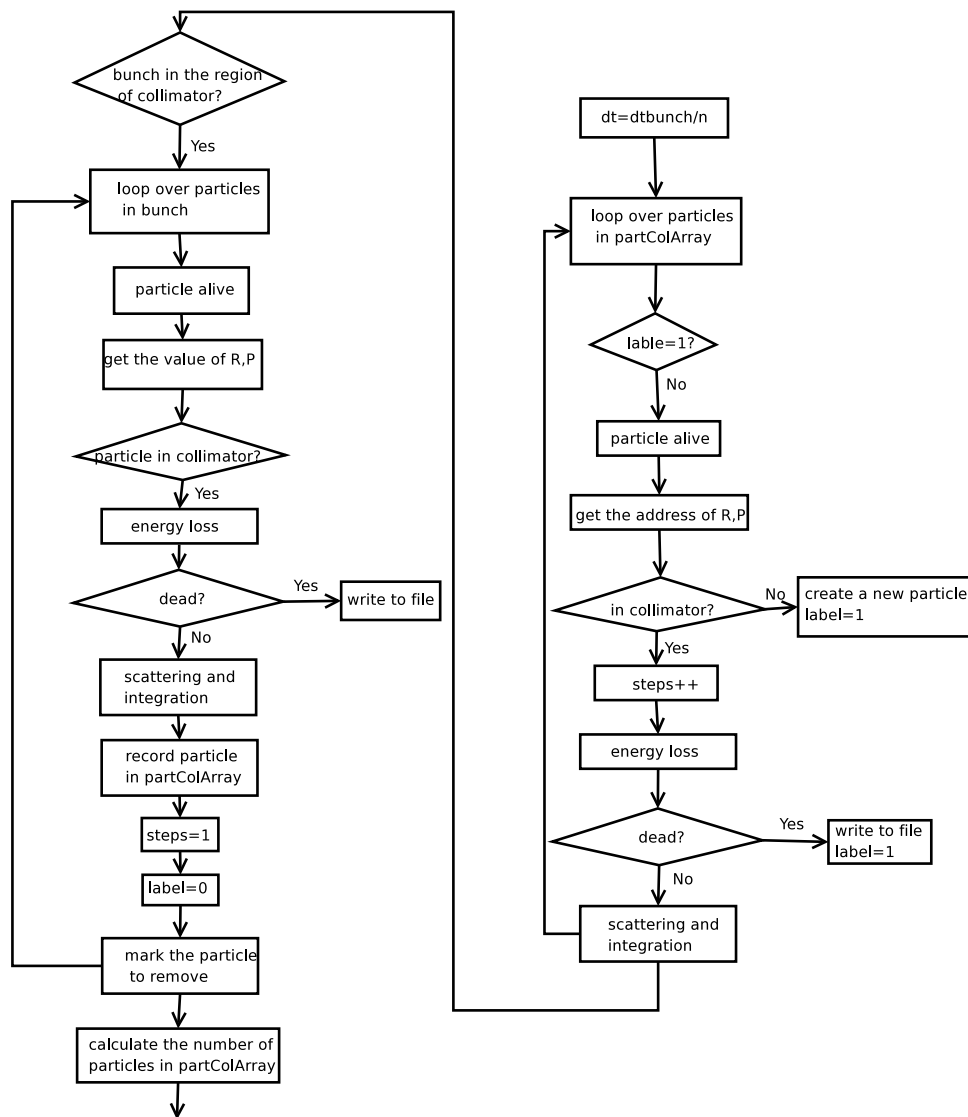


Figure 18.3: The diagram of CollimatorPhysics in OPAL.

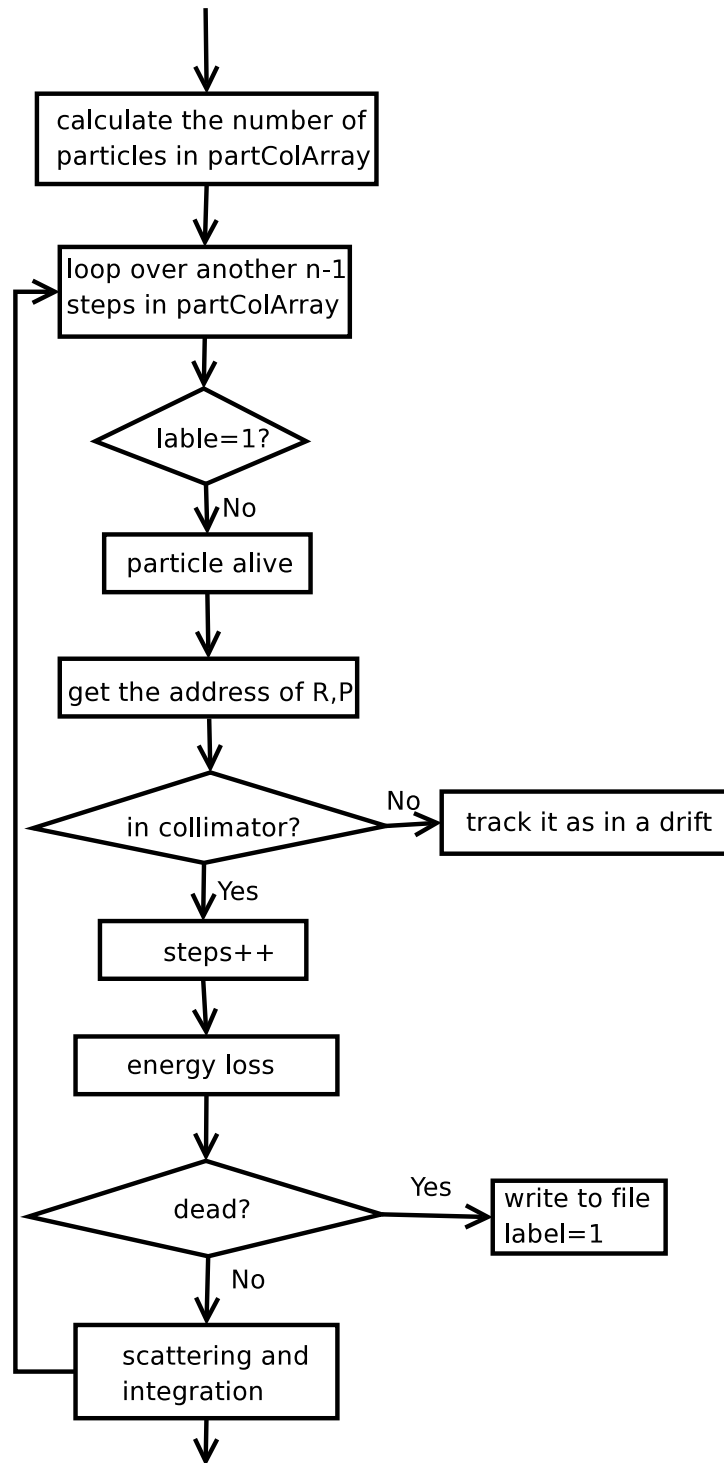


Figure 18.4: The diagram of CollimatorPhysics in OPAL(Continued).

18.3.1 The Substeps

Small step is needed in the routine of CollimatorPhysics.

If a large step is given in the main input file, in the file CollimatorPhysics.cpp, it is divided by a integer number n to make the stepsize using for the calculation of collimator physics less than $1.01\text{e-}12$ s. As shown by Fig. 18.3 and Fig. 18.4 in the previous section, first we track one step for the particles already in the collimator and the newcomers, then another $(n-1)$ steps to make sure the particles in the collimator experience the same time as the ones in the main bunch.

Now, if the particle leave the collimator during the $(n-1)$ steps, we track it as in a drift and put it back to the main bunch when finishing $(n-1)$ steps.

18.4 Available Materials in OPAL

Material	Z	A	ρ [g/cm ³]	X0 [g/cm ²]	A2	A3	A4	A5	OPAL Name
Aluminum	13	26.98	2.7	24.01	4.739	2766	164.5	2.023E-02	Aluminum
AluminumAl203	50	101.96	3.97	27.94	7.227	11210	386.4	4.474e-3	AluminumAl203
Copper	29	63.54	8.96	12.86	4.194	4649	81.13	2.242E-02	Copper
Graphite	6	12.0172	2.210	42.7	2.601	1701	1279	1.638E-02	Graphite
GraphiteR6710	6	12.0172	1.88	42.7	2.601	1701	1279	1.638E-02	GraphiteR6710
Titan	22	47.8	4.54	16.16	5.489	5260	651.1	8.930E-03	Titan
Air	7	14	0.0012	37.99	3.350	1683	1900	2.513E-02	Air
Kapton	6	12	1.4	39.95	2.601	1701	1279	1.638E-02	Kapton
Gold	79	197	19.3	6.46	5.458	7852	975.8	2.077E-02	Gold
Water	10	18	1	36.08	2.199	2393	2699	1.568E-02	Water
Mylar	6.702	12.88	1.4	39.95	3.35	1683	1900	2.513E-02	Mylar
Berilium	4	9.012	1.848	65.19	2.590	966.0	153.8	3.475E-02	Berilium
Molybdenum	42	95.94	10.22	9.8	7.248	9545	480.2	5.376E-03	Molybdenum

Table 18.1: List of materials with their parameters implemented in OPAL

18.5 Example of an Input File

```

examples/surfacephysics.in
KX1IPHYS: SurfacePhysics, TYPE="Collimator",MATERIAL="Copper";
KX2IPHYS: SurfacePhysics, TYPE="Collimator",MATERIAL="Graphite";
KX0I: ECollimator, L=0.09, ELEMEDGE=0.01, APERTURE={0.003,0.003},OUTFN="KX0I.h5",
SURFACEPHYSICS='KX1IPHYS';
FX5: Slit, L=0.09, ELEMEDGE=0.01, APERTURE={0.005,0.003},
SURFACEPHYSICS='KX2IPHYS';

```

```
FX16: Slit, L=0.09, ELEMEDGE=0.01, APERTURE={-0.005,-0.003},
      SURFACEPHYSICS='KX2IPHYS';
```

FX5 is a slit in x direction, the APERTURE is **POSITIVE**, the first value in APERTURE is the left part, the second value is the right part. FX16 is a slit in y direction, the APERTURE is **NEGATIVE**, the first value in APERTURE is the down part, the second value is the up part.

18.6 A Simple Test

A cold Gaussian beam with $\sigma_x = \sigma_y = 5$ mm. The position of the collimator is from 0.01 m to 0.1 m, the half aperture in y direction is 3 mm. Fig. 18.5 shows the trajectory of particles which are either absorbed or deflected by a copper slit. As a benchmark of the collimator model in OPAL, Fig. 18.6 shows the energy spectrum and angle deviation at $z=0.1$ m after an elliptic collimator.

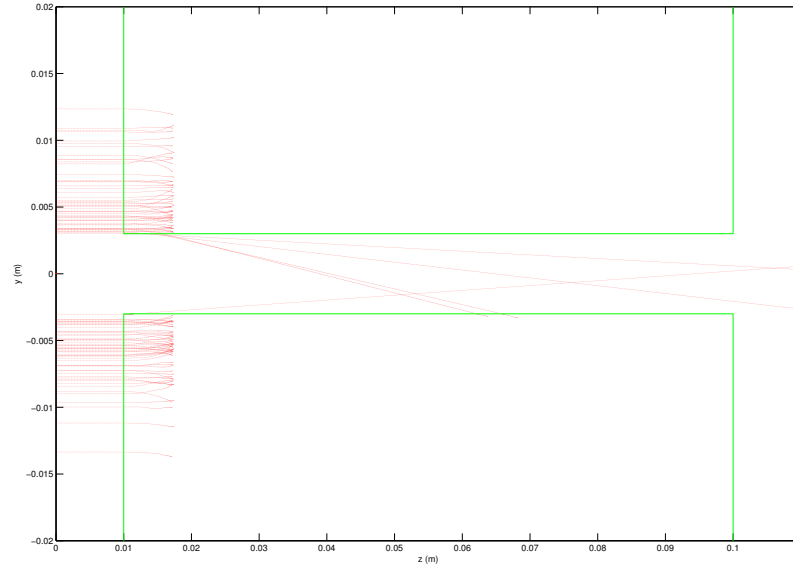
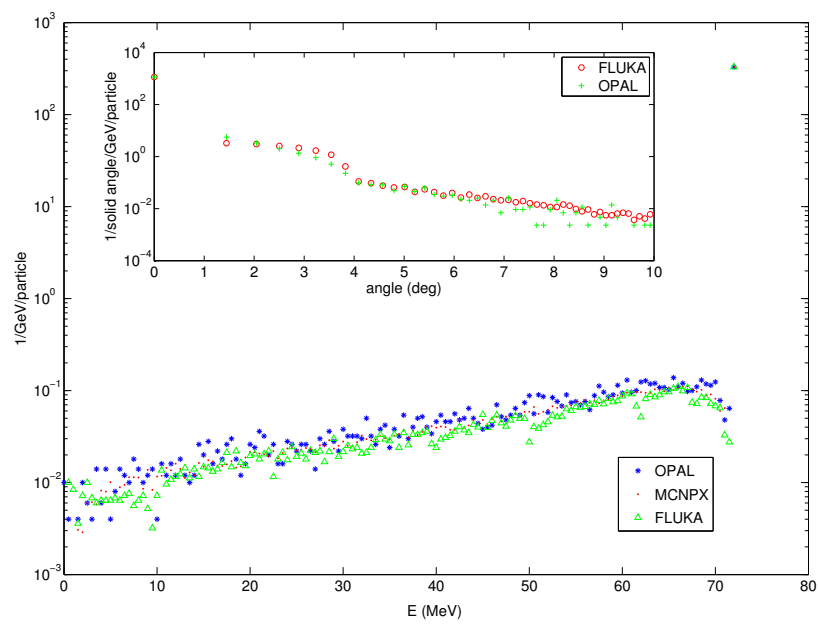


Figure 18.5: The passage of protons through the collimator.

Figure 18.6: The energy spectrum and scattering angle at $z=0.1$ m

Appendix A

Installation

OPAL and all its flavours are based on several packages which are all installable using `cmake` or the `configure-make-install` trilogy.

OPAL is also preinstalled on several HPC clusters including the Merlin cluster at PSI. The preinstalled version can be accessed using the module command:

```
module load opal
```

Due to some incompatibilities, between the Intel compiler and the Gnu libraries, you have to use GCC 4.5 in combination with Intel 12.1. Next we describe the installation process for the GNU 4.6.3 compiler.

A.1 Build and install OPAL on a Mac & Linux

A.1.1 Supporting Libraries

Several libraries and tools must be present before starting with the actual OPAL installation process described in (A.1.3). The following packages are maybe already installed. Please check the versions carefully and do not use older ones. For a Linux installation you can skip `macport` and `Xcode` related software.

- `macport` from www.macports.org
- `Xcode` <http://developer.apple.com/TOOLS/Xcode/>
- `automake-1.10.2`
- `autoconf-2.64`
- `libtool-2.2`
- `cmake-2.8.4`

A.1.2 Environment Variables

Assuming OPAL resides in `$HOME/svnwork`, the following environment variables must be set accordingly:

```
export OPAL_ROOT=$HOME/svnwork/OPAL/  
export CLASSIC_ROOT=$OPAL_ROOT/classic/5.0/
```

Build/Install gcc (4.6.3) Mac

```
sudo port install gcc46
```

Build/Install gsl (1.14) Mac

```
sudo port install gsl
```

Build/Install OpenMPI (openmpi-1.4.3) Mac & Linux

```
CC=gcc CXX=g++ F77=gfortran ./configure
```

Build/Install HDF5 1.8.7 Mac & Linux

```
./configure --enable-parallel --prefix=/usr/local
```

```
Libraries have been installed in:
    /usr/local/lib
```

```
If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the '-LLIBDIR'
flag during linking and do at least one of the following:
    - add LIBDIR to the 'DYLD_LIBRARY_PATH' environment variable
      during execution
```

```
See any operating system documentation about shared libraries
for more information, such as the ld(1) and ld.so(8) manual
pages.
```

H5hut (1.99.12) Mac & Linux

The tarball can be found at:

amas.psi.ch/H5hut/wiki/H5hutDownload

Now we can build and install the package:

```
./autogen.sh
./configure --enable-parallel --prefix=prefixdir
make
make install
```

Add to your `.bashrc`:

```
export C_INCLUDE_PATH=prefixdir/include:$C_INCLUDE_PATH
export CPLUS_INCLUDE_PATH=prefixdir/include:$CPLUS_INCLUDE_PATH
export LIBRARY_PATH=prefixdir/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=prefixdir/lib:$LD_LIBRARY_PATH
```

With gitorius:

1. Login to `gitorious.psi.ch` with your PSI username/password
2. upload your public key(s)
3. clone repo: `git clone git@gitorious.psi.ch:h5hut/src.git`

A.1.3 Installing OPAL

The following svn checkout ¹

```
cd $OPAL_ROOT
svn+ssh://savannah02.psi.ch/repos/opal/src .
```

will get you the trunk of the repository. Now install OPAL ²

```
mkdir build
cd build
CXX=mpicxx cmake -DCMAKE_BUILD_TYPE=RELEASE $OPAL_ROOT
```

Use `-DCMAKE_BUILD_TYPE=DEBUG` to enable `-g`.

A.2 Cray XE6 Installation – outdated

These notes are for installing OPAL on Hopper at the National Energy Research Scientific Computing Center (NERSC). NERSC is an open science computing center sponsored by the U.S. Department of Energy and located at Lawrence Berkeley National Laboratory (LBL). Hopper is a Cray XE6 system and, at the time of this writing, is their premier super computer.

The instructions below work, but you may want to modify them depending on where you prefer various codes and libraries to be located or if you are on a similar Cray system.

A.2.1 .bash_profile.ext File

Edit the `.bash_profile.ext` file in your home directory. Add the following lines.

```
if [ $NERSC_HOST == "hopper" ]
then
  export PATH=$PATH:$HOME/hopper/bin
  # Module files.
  module load cmake
  module load gsl
  module load hdf5-parallel
  module load zlib
  module swap PrgEnv-pgi PrgEnv-gnu
fi

source $HOME/.bashrc
```

A.2.2 .bashrc.ext File

Edit the `.bashrc.ext` file in your home directory. Add the following lines.

```
if [ $NERSC_HOST == "hopper" ]
then
  # For OPAL.
  export HDF5_INCLUDE_PATH=$CRAY_HDF5_DIR/hdf5-parallel-gnu/include
  export HDF5_LIBRARY_PATH=$CRAY_HDF5_DIR/hdf5-parallel-gnu/lib
  export H5hut=$HOME/hopper/extlib/H5hut-1.99.11
```

¹If you can not checkout the sources send an email to andreas.adelmann@psi.ch.

²We encountered in some cases a problem when the build directory is under \$OPAL_ROOT

```

export GSL_PREFIX=$GSL_DIR
export OPAL_ROOT=$HOME/svnwork/OPAL
export CLASSIC_ROOT=$OPAL_ROOT/classic/5.0
fi

```

These environment variables need to be set so we can build OPAL. When you are done, issue the command:

```
source $HOME/.bash_profile
```

A.2.3 OPAL

To install OPAL you will need to perform the following steps:

1. Install H5hut.
2. Install OPAL.

Install H5hut

1. `mkdir -p hopper/extlib`
2. `mkdir svnwork`
3. `cd $HOME/svnwork`
4. Get the source code:

```
\url{svn+ssh://savannah02.psi.ch/repos/H5hut/src/tags/1.99.11 H5hut-1.99.11}
```

5. `cd H5hut-1.99.11`
6. `./autogen.h`

```
export HDF5ROOT=$CRAY_HDF5_DIR/hdf5-parallel-gnu
```

7. `./configure --prefix=$HOME/hopper/extlib/H5hut-1.99.9 \`
`--enable-parallel CFLAGS=-std=c99 CC=cc`
-

8. `make`
9. `make install`

With gitorius:

1. Login to `gitorious.psi.ch` with your PSI username/password
2. upload your public key(s)
3. clone repo: `git clone git@gitorious.psi.ch:h5hut/src.git`

Install OPAL

1. `cd`
2. `cd svnwork`
3. Get the source code:

```
svn co --username username \  
svn+ssh://savannah02.psi.ch/repos/opal/src OPAL
```

4. `cd $OPAL_ROOT`

5. `CXX=CC cmake $OPAL_ROOT`

6. `make\verb`

7. `mkdir $HOME/hopper/bin`

8. `cp src/opal $HOME/hopper/bin`

A.3 Build unit tests

Some portions of OPAL can be tested using *unit tests*. Unit tests are implemented to ensure that a particular unit e.g. a function or class works as intended. OPAL uses the google testing *gtest* framework to support unit tests. Documentation can be found at <https://code.google.com/p/googletest/w/list>.

The OPAL unit tests are by default not built. To build the unit tests, first gtest must be installed.

```
cd ${OPAL_ROOT}/src/unit_tests/gtest  
bash bin/install_gtest.bash
```

This will check that you have the required packages to build gtest, download the gtest source code from the google website and attempt to build it. Built files should be placed into `trunk/src/unit_tests/gtest/`.

Next modify the cmake files to flag that the unit tests need to be built.

```
cd ${OPAL_ROOT}/  
cmake -DBUILD_OPAL_UNIT_TESTS=1  
make
```

If all goes well, the unit tests should build. To execute the tests, do

```
cd ${OPAL_ROOT}/  
./src/opal_unit_tests
```

You should get output that looks like e.g.

```
[=====] Running 47 tests from 8 test cases.
[-----] Global test environment set-up.
[-----] 6 tests from OpalRingSectionTest
[ RUN      ] OpalRingSectionTest.TestConstructDestruct
[          OK ] OpalRingSectionTest.TestConstructDestruct (0 ms)
<snip>
[ RUN      ] PolynomialTimeDependenceTest.TDMapNameLookupTest
[          OK ] PolynomialTimeDependenceTest.TDMapNameLookupTest (0 ms)
[-----] 3 tests from PolynomialTimeDependenceTest (0 ms total)

[-----] Global test environment tear-down
[=====] 47 tests from 8 test cases ran. (25 ms total)
[ PASSED  ] 47 tests.

YOU HAVE 1 DISABLED TEST
```

If the output does not end in PASSED then something failed. Review the test output to find the test failure.

Adding a unit test

To add a unit test to the OPAL framework, either edit an existing file or add a new file. In general, there should be one unit test per pair of header, source files. The directory structure reflects OPAL directory structure. Source code that is in `${OPAL_ROOT}/classic/5.0/src/<dir>/<Filename.h,cpp>` should have corresponding unit test files in `${OPAL_ROOT}/src/unit_tests/classic_src/<dir>/<FilenameTest.cpp>`. Source code that is in `${OPAL_ROOT}/src/<dir>/<Filename.h,cpp>` should have corresponding unit test files in `${OPAL_ROOT}/src/unit_tests/opal_src/<dir>/<FilenameTest.cpp>`. To update the list of files used by cmake, do

```
touch ${OPAL_ROOT}/CMakeLists.txt
```

Then run make as usual.

A.4 Using pre-build Binaries

Pre-build binaries are available for SL Linux and Mac OS X at the following download page: <https://amas.psi.ch/OPAL/wiki/Downloads>.

A.5 Enabling the Iterative Solvers for Space-Charge Calculation

Iterative solvers are used to solve the Poisson's equation for the electrostatic potential on three-dimensional domains [52]. The multigrid preconditioner and iterative solver are implemented using Trilinos framework. It is essential to install Trilinos software framework to use the iterative solvers in OPAL.

Trilinos is also preinstalled on several HPC clusters including the Merlin cluster at PSI. The preinstalled version can be accessed using the module command:

```
module load trilinos
```

Please note: The Multigrid space charge solver is not yet capable of emitting a beam from the cathode. You are advised to use the FFT space charge solver for studies with particle emission.

Trilinos Package

If Trilinos version (>11.0) is not available on your system, download it from the Trilinos webpage <http://trilinos.sandia.gov/download>³. CMake build system is used to configure, build and install Trilinos. Instructions for building Trilinos is available in

<http://trilinos.sandia.gov/TrilinosBuildQuickRef.html>

To build OPAL with SAAMG_SOLVER or AMR_SOLVER, Trilinos packages, which are needed to be enabled are:

- epetra and epetraext
- ml and ml_parmetis3x
- amesos and amesos-superludist
- ifpack
- teuchos and teuchos-extended
- aztecco and aztecco-teuchos
- galeri
- belos

and Third-Party Library (TPL) support

- ParMETIS (-D TPL_ENABLE_ParMETIS:BOOL=ON)
- BLAS (-D TPL_ENABLE_BLAS:BOOL=ON)
- LAPACK (-D TPL_ENABLE_LAPACK:BOOL=ON)
- MPI (-D TPL_ENABLE_MPI:BOOL=ON)

To enable a given TPL, the path to this package's header include and library directories should be specified when building Trilinos. The TPL libraries such as libparmetis.a libmetis.a libblas.a liblapacke.a liblapack.a are needed to use MultiGrid (SAAMG) and Adaptive Mesh Refinement (AMR) solvers.

A.5.1 Environment Variables

Assuming Trilinos package is installed in \$TRILINOS_PREFIX then the following environment variables must be set accordingly.

TRILINOS_DIR should point the path to the base installation directory of Trilinos

TRILINOS_INCLUDE_DIR should point the path to Trilinos' include directory

TRILINOS_LIBRARY_DIR should point the path to Trilinos' library directory

If you are using preinstalled version of Trilinos on Merlin cluster at PSI, the environment variables are automatically set so you don't have to set these yourself.

```
export TRILINOS_DIR=$TRILINOS_PREFIX
export TRILINOS_INCLUDE_DIR=$TRILINOS_DIR/include
export TRILINOS_LIBRARY_DIR=$TRILINOS_DIR/lib
```

³<http://trilinos.sandia.gov>

A.5.2 Build OPAL with SAAMG_SOLVER enabled

You can enable the solver with

`-DENABLE_SAAMG_SOLVER=TRUE` using `cmake` and make sure the environment variables A.5.1 are set up already before OPAL installation.

```
cd $OPAL_ROOT
mkdir build
cd build
cmake -DENABLE_SAAMG_SOLVER=TRUE $OPAL_ROOT
make
```

Notes: The members should appear in the initializer list in the same order as they appear in the class or turn `-Wno-reorder`. C++11 language features are available in gcc 4.3 and later versions. To enable this support, add `-std=c++0x` to your command line if you are using gcc 4.6 OR `-std=c++11` if you are using gcc 4.7 and later.

A.5.3 Build OPAL with AMR_SOLVER enabled

BoxLib⁴ software framework is used in OPAL to add the adaptive mesh refinement (AMR) technique. You can enable AMR solver with

`-DENABLE_AMR_SOLVER=TRUE` using `cmake`. Please make sure the following environment variables are set and `CCSE_INCLUDE_DIRS` and `CCSE_LIB_DIR` point the path to the include and library directories of BoxLib.

```
export BOXLIB_HOME=$HOME/svnwork/BoxLib
export BOXLIB_INSTALL_PREFIX=$BOXLIB_HOME/Build
export CCSE_INCLUDE_DIRS=$BOXLIB_INSTALL_PREFIX/include
export CCSE_LIBRARY_DIR=$BOXLIB_INSTALL_PREFIX/lib
```

```
cd $OPAL_ROOT
mkdir build
cd build
cmake -DENABLE_AMR_SOLVER=TRUE $OPAL_ROOT
make
```

Notes: The members should appear in the initializer list in the same order as they appear in the class or turn `-Wno-reorder`.

BoxLib Package

You can download BoxLib from the LBNL CCSE server by typing the following command `git clone`

```
cd $HOME/svnwork
git clone https://ccse.lbl.gov/pub/Downloads/BoxLib.git
```

BoxLib based code designed to run in parallel using MPI, check the following environment variable before start building BoxLib.

```
export BOXLIB_HOME=$HOME/svnwork/BoxLib
export BOXLIB_INSTALL_PREFIX=$BOXLIB_HOME/Build
export MPIHOME=/opt/openmpi/1.4.5/gcc-mp/4.7
```

Build and install as follows:

⁴<https://ccse.lbl.gov/BoxLib>

```

cd BoxLib
mkdir Build
cmake -DBL_SPACEDIM=3 -DENABLE_MPI=1 -DENABLE_OpenMP=0 -DBL_USE_PARTICLES=1
      -DCMAKE_INSTALL_PREFIX=$BOXLIB_INSTALL_PREFIX $BOXLIB_HOME
make
make install
export CCSE_INCLUDE_DIRS=$BOXLIB_INSTALL_PREFIX/include
export CCSE_LIBRARY_DIR=$BOXLIB_INSTALL_PREFIX/lib

```

Now you are ready to build OPAL with AMR support.

A.6 Debug Flags

Table A.1: Debug flags.

Name	Description	Default
DBG_SCALARFIELD	dumps scalar potential on the grid	not set
DBG_STENCIL	dumps stencil (SAAMG solver) to a Matlab readable file	not set
DBG_CSR	dump information regarding the 1D CSR calculation	not set

DBG_SCALARFIELD dumps the field to a file called rho_scalar. The structure of the data can be deduced from the following Matlab script:

```

function scalffield(RHO)

rhosize=size(RHO)
for i=1:rhosize(1)
    x = RHO(i,1);
    y = RHO(i,2);
    z = RHO(i,3);
    rhoxyz(y,z) = RHO(i,4);
    rhoxy(x,y) = RHO(i,4);
    rhoxz(x,z) = RHO(i,4);
    rho(x,y,z) = RHO(i,4);
end

```

DBG_STENCIL dumps the discretization stencil to a file (A.dat). The following Matlab code will read and store the sparse matrix in the variable 'A'.

```

load A.dat;
A = spconvert(A);

```

DBG_CSR dumps a text file of the calculated, 1D CSR field. The first line gives the average position of the beam bunch. Subsequent lines list z position of longitudinal mesh (with respect to the head of the beam bunch), E_z , line density and the derivative of the line density. Note that currently the line density derivative needs to be scaled by the inverse of the mesh spacing to get the correct value. The CSR field is dumped at each time step of the calculation. Each text file is named "Bend Name" (from input file) + "-CSRWake" + "time step number in that bend (starting from 1)" + ".txt".

A.7 OPAL as a Library

An OPAL library can be build by specifying `-DBUILD_LIBOPAL` in the cmake process. The OPAL libraey is currently used in the opt-pilot, a multi-objective optimization package [65].

A.8 Examples

When checking out the OPAL framework you will find the *tests* directory and moreover a subdirectory called *RegressionTests*. There several input files can be found which are run every day to check the validity of the current version of OPAL. This is a good starting-point to learn how to model accelerators with the various flavors of OPAL. More examples will be given in subsequent chapters, enjoy!

Appendix B

OPAL Language Syntax

Words in *italic font* are syntactic entities, and characters in `monospaced font` must be entered as shown. Comments are given in **bold font**.

Statements:

<i>comment</i>	:	<code>// anything-except-newline</code>
		<code>/* anything-except-*/ */</code>
<i>identifier</i>	:	<code>[a-zA-Z][a-zA-Z0-9-]</code>
<i>integer</i>	:	<code>[0-9]+</code>
<i>string</i>	:	<code>' anything-except-single-quote'</code>
		<code>"anything-except-double-quote"</code>
<i>command</i>	:	<code>keyword attribute-list</code>
		<code>label : keyword attribute-list</code>
<i>keyword</i>	:	<i>identifier</i>
<i>label</i>	:	<i>identifier</i>
<i>attribute-list</i>	:	<i>empty</i>
		<i>attribute-list , attribute</i>

<i>attribute</i>	:	<i>attribute-name</i> // only for logical attribute
		<i>attribute-name</i> = <i>attribute-value</i>
		// expression evaluated
		<i>attribute-name</i> := <i>attribute-value</i>
		// expression retained
<i>attribute-name</i>	:	<i>identifier</i>
<i>attribute-value</i>	:	<i>string-expression</i>
		<i>logical-expression</i>
		<i>real-expression</i>
		<i>array-expression</i>
		<i>constraint</i>
		<i>variable-reference</i>
		<i>place</i>
		<i>range</i>
		<i>token-list</i>
		<i>token-list-array</i>
		<i>regular-expression</i>

Real expressions:

<i>real-expression</i>	:	<i>real-term</i>
		+ <i>real-term</i>
		– <i>real-term</i>
		<i>real-expression</i> + <i>real-term</i>
		<i>real-expression</i> – <i>real-term</i>
<i>real-term</i>	:	<i>real-factor</i>
		<i>real-term</i> * <i>real-factor</i>
		<i>real-term</i> / <i>real-factor</i>
<i>real-factor</i>	:	<i>real-primary</i>
		<i>real-factor</i> ^ <i>real-primary</i>
<i>real-primary</i>	:	<i>real-literal</i>
		<i>symbolic-constant</i>
		#
		<i>real-name</i>
		<i>array</i> [<i>index</i>]
		<i>object-name</i> -> <i>real-attribute</i>
		<i>object-name</i> -> <i>array-attribute</i> [<i>index</i>]
		<i>table-reference</i>
		<i>real-function</i> ()
		<i>real-function</i> (<i>real-expression</i>)
		<i>real-function</i> (<i>real-expression</i> , <i>real-expression</i>)
		<i>function-of-array</i> (<i>array-expression</i>)
		(<i>real-expression</i>)

<i>real-function</i>	:	RANF
		GAUSS
		ABS
		TRUNC
		ROUND
		FLOOR
		CEIL
		SIGN
		SQRT
		LOG
		EXP
		SIN
		COS
		ABS
		TAN
		ASIN
		ACOS
		ATAN
		ATAN2
		MAX
		MIN
		MOD
		POW

<i>symbolic-constant</i>	:	PI
		TWOPI
		DEGRAD
		RADDEG
		E
		EMASS
		PMASS
		HMMASS
		UMASS
		CMASS
		MMASS
		DMASS
		XEMASS
		CLIGHT
		<i>real-name</i>
<i>real-name</i>	:	<i>identifier</i>
<i>object-name</i>	:	<i>identifier</i>
<i>table-name</i>	:	<i>identifier</i>
<i>column-name</i>	:	<i>identifier</i>

Logical expressions:

<i>logical-expression</i>	:	<i>and-expression</i>
		<i>logical-expression</i> <i>and-expression</i>
<i>and-expression</i>	:	<i>relation</i>
		<i>and-expression</i> && <i>relation</i>

<i>relation</i>	:	<i>logical-name</i>
		TRUE
		FALSE
		<i>real-expression relation-operator real-expression</i>
<i>logical-name</i>	:	<i>identifier</i>
<i>relation-operator</i>	:	==
		!=
		<
		>
		>=
		<=

Logical variables:

<i>logical-prefix</i>	:	BOOL
		BOOL CONST
<i>logical-definition</i>	:	<i>logical-prefix logical-name</i> = <i>logical-expression</i>
		// expression evaluated
		<i>logical-prefix logical-name</i> := <i>logical-expression</i>
		// expression retained

String expressions:

<i>string-expression</i>	:	<i>string</i>
		<i>identifier</i> // taken as a string
		<i>string-expression</i> & <i>string</i>

String constants:

```

string-prefix          :  STRING

string-definition     :  string-prefix string-name = string-expression

                        // expression evaluated

                        |  string-prefix string-name := string-expression

                        // expression retained

```

Real array expressions:

```

array-expression      :  array-term

                        |  + array-term

                        |  − array-term

                        |  array-expression + array-term

                        |  array-expression − array-term

array-term             :  array-factor

                        |  array-term * array-factor

                        |  array-term / array-factor

array-factor           :  array-primary

                        |  array-factor ^ array-primary

array-primary          :  { array-literal }

                        |  array-reference

                        |  real-function ( array-expression )

                        |  ( array-expression )

array-literal          :  real-expression

                        |  array-literal , real expression

```

array-reference : *array-name*
 | *object-name* -> *array-attribute*

array-name : *identifier*

Real array definitions:

array-prefix : REAL VECTOR

array-definition : *array-prefix array-name* = *array-expression*
 | *array-prefix array-name* := *array-expression*

Constraints:

constraint : *array-expression constraint-operator array-expression*

constraint-operator : ==
 | <
 | >

Variable references:

variable-reference : *real-name*
 | *object-name* -> *attribute-name*

Token lists:

token-list : *anything-except-comma*

token-list-array : *token-list*
 | *token-list-array* , *token-list*

Regular expressions:

regular-expression : "UNIX-regular-expression"

Appendix C

OPAL-T Field Maps

C.1 Introduction

In this chapter details of the different types of field maps in OPAL-T are presented. OPAL-T can use many different types of field maps input in several different file formats. What types of maps are supported and in what format has tended to be a function mostly of what developers have needed, and to a lesser extent what users have asked for. The list below shows all field maps that are currently supported and also field maps that are not yet supported, but on the list of things to do when we get a chance.

C.2 Comments in Field Maps

The possibility to add comments (almost) everywhere in field map files is common to all field maps. Comments are initiated by a # and contain the rest of a line. Comments are accepted at the beginning of the file, between the lines and at the end of a line. If in the following sections two values are shown on one line then they have to be on the same line. They should not be separated by a comment and, consequently, be on different lines. Three examples of valid comments:

```
# This is valid a comment
1DMagnetoStatic 40 # This is another valid comment
-60.0 60.0 9999
# and this is also a valid comment
0.0 2.0 199
```

The following examples will break the parsing of the field maps:

```
1DMagnetoStatic # This is an invalid comment
40
-60.0 60.0 # This is another invalid comment # 9999
0.0 2.0 199
```

C.3 Field Map Warnings and Errors

If OPAL-T encounters an error while parsing a field map it disables the corresponding element, outputs a warning message and continues the simulation. The following messages may be output:

```
***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELD MAP file.t7.
```

```

There are only 10003 lines in the file, expecting more.
Please check the section about field maps in the user manual.
*****

```

In this example there is something wrong with the number of grid spacings provided in the header of the file. Make sure that you provide the number of grid **spacings** and not the number of grid **points**! The two numbers always differ by 1.

```

***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELD MAP file.t7.
There are too many lines in the file, expecting only 10003 lines.
Please check the section about field maps in the user manual.
*****

```

Again there seems to be something wrong with the number of grid spacings provided in the header. In this example OPAL-T found more lines than it expected. Note that comments and empty lines at the end of a file are ignored such that **they don't cause** this warning.

```

***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELD MAP file.t7.
_error_msg_
expecting: '_expecting_' on line 3,
found instead: '_found_'.
*****

```

Where `_error_msg_` is either

Didn't find enough values!	If OPAL-T expects more values on this line.
Found more values than expected!	If OPAL-T expects less values on this line.
Found wrong type of values!	If OPAL-T found e.g. characters instead of an integer number.

`'_expecting_'` is replaced by the types of values OPAL-T expects on the line. E.g. it could be replaced by `'double double int'`. Finally `'_found_'` is replaced by the actual content of the line without any comment possibly following the values. If line 3 of a file consists of `'-60.0 60.0 # This is an other invalid comment # 9999'` OPAL-T will output `'-60.0 60.0'`.

```

***** W A R N I N G *****
DISABLING FIELD MAP file.t7 SINCE FILE COULDN'T BE FOUND!
*****

```

This warning could be issued if the filename is mistyped or otherwise if the file couldn't be read.

```

***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELDMAP file.t7.
Could not determine the file type.
Please check the section about field maps in the user manual.
*****

```

In this case OPAL-T didn't recognise the string of characters which identify the type of field map stored in the file. For one-dimensional field maps an other warning may be issued:

```

* ***** W A R N I N G *****
* IT SEEMS THAT YOU USE TOO FEW FOURIER COMPONENTS TO SUFFICIENTLY WELL      *
* RESOLVE THE FIELD MAP 'file.T7'.                                           *
* PLEASE INCREASE THE NUMBER OF FOURIER COMPONENTS!                         *
* The ratio (f_i - F_i)^2 / F_i^2 is 0.019685 and                            *
* the ratio (max_i(|f_i - F_i|) / max_i(|F_i|)) is 0.019023.                 *
* Here F_i is the field as in the field map and f_i is the reconstructed      *
* field.                                                                      *
* The lower limit for the two ratios is 1e-2                                 *
* *****

```

This warning is issued when the low pass filter that is applied to the field sampling uses too few Fourier coefficients. In this case increase the number of Fourier coefficients, see the next section for details. The relevant criterions are that

$$\frac{\sum_{i=1}^N (F_{z,i} - \tilde{F}_{z,i})^2}{\sum_{i=1}^N F_{z,i}^2} \leq 0.01,$$

and

$$\frac{\max_i |F_{z,i} - \tilde{F}_{z,i}|}{\max_i |F_{z,i}|} \leq 0.01,$$

where $F_{z,i}$ is the field sampling as in the file and $\tilde{F}_{z,i}$ is the one-dimensional field reconstructed from the result received after applying the low pass filter.

C.4 Types and Format

Field maps in OPAL-T come in three basic types:

1. 2D or 3D field map. For this type of map, a field is specified on a grid and linear interpolation is used to find field values at intermediate points.
2. 1D on axis field map. For this type of map, on on axis field profile is specified. OPAL-T calculates a Fourier series from this profile and then uses the 1st, 2nd and 3rd derivatives of the series to compute the off-axis field values. (This type of field is very smooth numerically, but can be inaccurate far from the field axis.) Only a few (user specified) terms from the Fourier series are used.
3. Enge function [44] field map. This type of field map uses Enge functions to describe the fringe fields of a magnet. Currently, this is only used for RBEND and SBEND elements (see 8.4.1 and 8.4.2).

It is important to note that in all cases, the input field map will be normalized so that the peak field magnitude value on axis is equal to either 1 MV/m in the case of electric field maps (static or dynamic), or 1 T in the case of magnetic field maps. (The sign of the values from the field map are preserved.) Therefore, the field multiplier for the map in your simulation will be the peak field value on axis in those respective units.

Depending on the field map type, OPAL-T uses different length units (either cm or meters). This is due to the origin programs of the field maps used (e.g. Poisson/Superfish [43] uses cm). So be careful.

There are no required field extensions for any OPAL-T field map (e.g. .T7, .dat etc.). OPAL-T determines the type of field map by a string descriptor which is the first element on the first line of the file. Below we list the possible descriptors. (Note that we list all of the descriptors/field map types that we plan to eventually implement. Not all of them are, which is indicated in the description.)

1DElectroStatic

1D electrostatic field map. 1D field maps are described by the on-axis field.

Not implemented yet. A work around is to use a 1DDynamic field map with a very low frequency.

1DMagnetoStatic

1D magnetostatic field map. See C.7.

AstraMagnetoStatic

1D magnetostatic field map with possibly non-equidistant sampling. This file type is compatible with ASTRA field maps with small changes. See C.8.

1DDynamic

1D dynamic electromagnetic field map. See C.9.

AstraDynamic

1D dynamic electromagnetic field map with possibly non-equidistant sampling. This file type is compatible with ASTRA field maps with small changes. See C.10.

1DProfile1

This type of field map specifies the Enge functions (see [44]) for the entrance and exit fringe fields of a magnet (see ??). Currently this type of field map is only used by RBEND and SBEND elements (see 8.4.1 and 8.4.2). See C.11.

2DElectroStatic

2D electrostatic field map. 2D field maps are described by the electromagnetic field in one half-plane. See C.12.

2DMagnetoStatic

2D magnetostatic field map. Other than this descriptor at the head of the file, the format for this field map type is identical to the T7 file format as produced by Poisson [43]. See C.13.

2DDynamic

2D dynamic electromagnetic field map. Other than this descriptor at the head of the file, the format for this field map type is identical to the T7 field format as produced by Superfish [43]. See C.14.

3DElectroStatic

3D electrostatic field map.
Not implemented yet.

3DMagnetoStatic

3D magnetostatic field map.
Not implemented yet.

3DDynamic

3D dynamic electromagnetic field map. See C.15.

We will give examples and descriptions of each of the implemented field map types in the sections below.

C.5 Field Map Orientation

In the case of 2D and 3D field maps an additional string has to be provided describing the orientation of the field map.

For 2D field maps this can either be

XZ

if the primary direction is in z direction and the secondary in r direction.

ZX

if the primary direction is in r direction and the secondary in z direction.

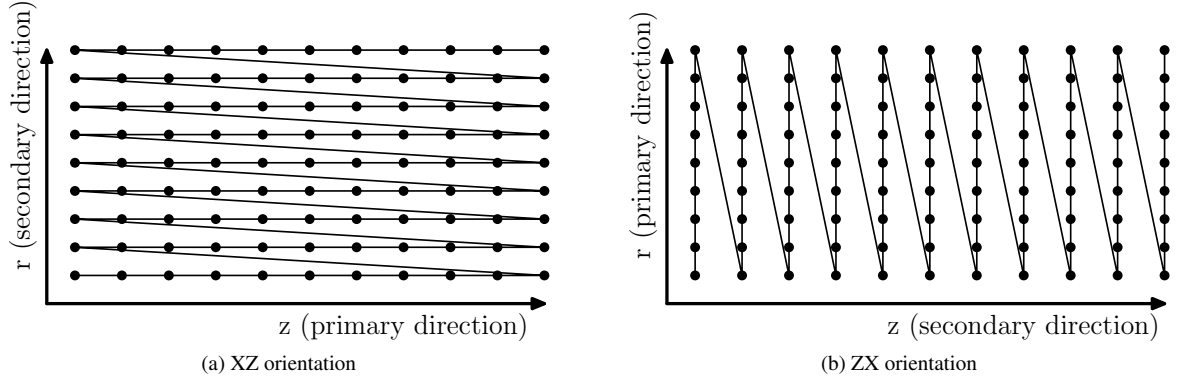


Figure C.1: Ordering of points for 2D field maps in T7 files

For 3D field maps this can be

XYZ

if the primary direction is in z direction, the secondary in y direction and the tertiary in x direction


Each line after the header corresponds to a grid point of the field map. This point can be referred to by two indices in the case of a 2D field map and three indices in the case of a 3D field map, respectively. Each column describes either E_z , E_r , B_z , B_r or H_ϕ in the 2D case and E_x , E_y , E_z , B_x , B_y , B_z in the 3D case.

By primary, secondary and tertiary direction is meant the following (see also Figure C.1a and Figure C.1b):

- The index of the primary direction increases the fastest, the index of the tertiary direction the slowest.
- The order of the columns is accordingly: if the z direction in an 2D electrostatic field map is the primary direction then E_z is on the first column, E_r on the second. For all other cases it's analogous.
- For the 2D dynamic case in XZ orientation there are four columns: E_z , E_r , $|E|$ (unused) and H_ϕ in that order. In the other orientations the first and the second columns are interchanged, but the third and fourth columns are unchanged.

C.6 FAST Attribute for 1D Field Maps

For some 1D field maps, there exists a boolean attribute, `FAST`, which can be used to speed up the calculation. When set to true (`FAST = TRUE`), OPAL-T will generate a 2D internal field map and then use bilinear interpolation to calculate field values during the simulation, rather than the generally slower Fourier coefficient technique. The caution here is that this can introduce unwanted numerical noise if you set the grid spacing too coarse for the 2D map.

 As a general warning: be wise when you choose the type of field map to be used! Figure C.2 shows three pictures of the longitudinal phase space after three gun simulations using different types of field maps. In the first picture, Figure C.2a, we used a 1DDynamic field map (see C.9) resulting in a smooth longitudinal distribution. In Figure C.2b we set the `FAST` attribute to true, resulting in some fine structure in the phase space due to the bilinear interpolation of the internally generated 2D field map. Finally, in the last figure (Figure C.2c), we generated directly a 2D field map from Superfish [43]. Here we could observe two different structures: first the fine structure, stemming from the bilinear interpolation, and secondly a much stronger structure of unknown origin, but presumably due to errors in the Superfish [43] interpolation algorithm.

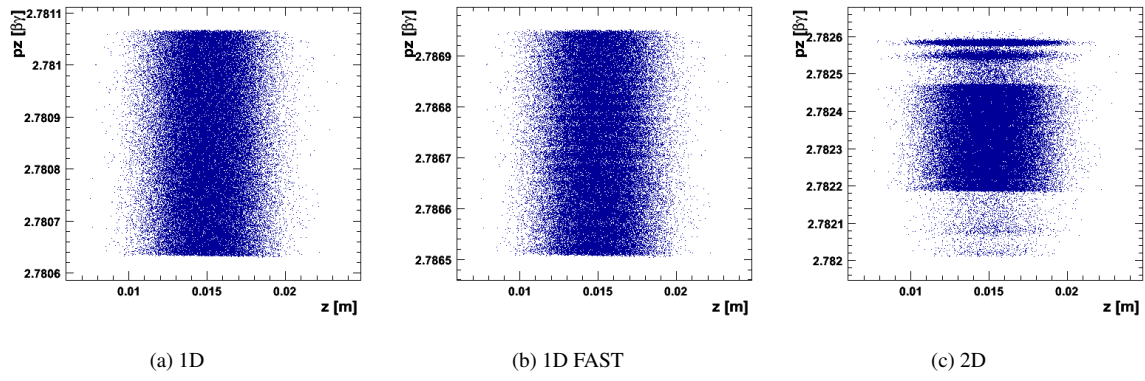


Figure C.2: The longitudinal phase space after a gun simulation using a 1D field map (on-axis field) of the gun, a 1D field map (on-axis field) of the gun in combination with the FAST switch, and a 2D field map of the gun generated by Superfish [43].

C.7 1DMagnetoStatic

```
1DMagnetoStatic 40
-60.0 60.0 9999
  0.0  2.0 199
  0.00000e+00
  4.36222e-06
  8.83270e-06
+ 9'994 lines
  1.32490e-05
  1.73710e-05
  2.18598e-05
```

Figure C.3: A 1D field map describing a magnetostatic field using 10000 grid points (9999 grid spacings) in the longitudinal direction. The field is non-negligible from -60.0 cm to $+60.0\text{ cm}$ relative to ELEMEDGE in the longitudinal direction. From the 10000 field values, 5000 complex Fourier coefficients are calculated. However, only 40 are kept when calculating field values during a simulation. OPAL-T normalizes the field values internally such that $\max(|B_{\text{on axis}}|) = 1.0\text{ T}$. If the FAST attribute is set to true in the input deck, a 2D field map is generated internally with 200 values in the radial direction, from 0 cm to 2 cm , for each longitudinal grid point.

Table C.1: Layout of a 1DMagnetoStatic field map file.

1DMagnetoStatic	N_{Fourier}	
z_{start} (in cm)	z_{end} (in cm)	N_z
r_{start} (in cm)	r_{end} (in cm)	N_r
$B_{z,1}$ (T)		
$B_{z,2}$ (T)		
.		
.		
.		
B_{z,N_z+1} (T)		

A 1DMagnetoStatic field map has the general form shown in Table C.1. The first three lines form the file header and tell OPAL-T how the field map data is being presented:

- Line 1** This tells OPAL-T what type of field file it is (1DMagnetoStatic) and how many Fourier coefficients to keep (N_{Fourier}) when doing field calculations.
- Line 2** This tells gives the extent of the field map (from z_{start} to z_{end}) relative to the ELEMEDGE of the field map, and how many grid spacings there are in the field map.
- Line 3** If one sets FAST = TRUE for the field map, this tells OPAL-T the radial extent of the internally generated 2D field map. Otherwise this line is ignored. (Although it must always be present.)

The lines following the header give the 1D field map grid values from 1 to $N_z + 1$. From these, $N_z/2$ complex Fourier coefficients are calculated, of which only N_{Fourier} are used when finding field values during the simulation.

Figure C.3 gives an example of a `1DMagnetoStatic` field file.

C.8 AstraMagnetostatic

```
AstraMagnetostatic 40
-3.0000000e-01 0.0000000e+00
-2.9800000e-01 2.9075045e-05
-2.9600000e-01 5.9367702e-05
-2.9400000e-01 9.0866460e-05
-2.9200000e-01 1.2374798e-04
-2.9000000e-01 1.5799850e-04
. . . .
2.9000000e-01 1.5799850e-04
2.9200000e-01 1.2374798e-04
2.9400000e-01 9.0866460e-05
2.9600000e-01 5.9367702e-05
2.9800000e-01 2.9075045e-05
3.0000000e-01 0.0000000e+00
```

Figure C.4: A 1D field map describing a magnetostatic field using N non-equidistant grid points in the longitudinal direction. From these values N equidistant field values are computed from which in turn $N/2$ complex Fourier coefficients are calculated. In this example only 40 Fourier coefficients are kept when calculating field values during a simulation. The z -position of each field sampling is in the 1st column (in meters), the corresponding longitudinal on-axis magnetic field amplitude is in the 2nd column. As with the 1DMagnetoStatic (see C.7) field maps, OPAL-T normalizes the field values to $\max(|B_{\text{on axis}}|) = 1.0$ T. In the header only the first line is headed since the information on the longitudinal dimension is contained in the first column of the data. (OPAL-T does not provide a FAST version of this map type.)

Table C.2: Layout of an AstraMagnetostatic field map file.

AstraMagnetostatic	N_{Fourier}
z_1 (in meters)	$B_{z,1}$ (T)
z_2 (in meters)	$B_{z,s}$ (T)
.	
.	
.	
z_N (in meters)	$B_{z,N}$ (T)

An AstraMagnetostatic field map has the general form shown in Table C.2. The first line forms the file header and tells OPAL-T how the field map data is being presented:

Line 1 This tells OPAL-T what type of field file it is (AstraMagnetostatic) and how many Fourier coefficients to keep (N_{Fourier}) when doing field calculations.

The lines following the header gives N non-equidistant field values and their corresponding z positions (relative to ELEMEDGE). From these, OPAL-T will use cubic spline interpolation to find N equidistant field values within

the range defined by the z positions. From these equidistant field values, $N/2$ complex Fourier coefficients are calculated, of which only N_{Fourier} are used when finding field values during the simulation.

Figure C.4 gives an example of an `AstraMagnetostatic` field file.

C.9 1DDynamic

```

1DDynamic 40
-3.0 57.0 4999
1498.953425154
0.0 2.0 199
  0.00000e+00
  4.36222e-06
  8.83270e-06
+ 4'994 lines
  1.32490e-05
  1.73710e-05
  2.18598e-05

```

Figure C.5: A 1D field map describing a dynamic field using 5000 grid points (4999 grid spacings) in the longitudinal direction. The field is non-negligible from -3.0 cm to 57.0 cm relative to ELEMEDGE in the longitudinal direction. The field frequency is $1498.953425154\text{ MHz}$. From the 5000 field values, 2500 complex Fourier coefficients are calculated. However, only 40 are kept when calculating field values during the simulation. OPAL-T normalizes the field values internally such that $\max(|E_{onaxis}|) = 1\text{ MV/m}$. If the FAST switch is set to true in the input deck, a 2D field map is generated internally with 200 values in the radial direction, from 0.0 cm to 2.0 cm , for each longitudinal grid point.

Table C.3: Layout of a 1DDynamic field map file.

1DDynamic	N_{Fourier}	
z_{start} (in cm)	z_{end} (in cm)	N_z
Frequency (in MHz)		
r_{start} (in cm)	r_{end} (in cm)	N_r
$E_{z,1}$ (MV/m)		
$E_{z,2}$ (MV/m)		
.		
.		
.		
E_{z,N_z+1} (MV/m)		

A 1DDynamic field map has the general form shown in Table C.3. The first four lines form the file header and tell OPAL-T how the field map data is being presented:

Line 1 This tells OPAL-T what type of field file it is (1DDynamic) and how many Fourier coefficients to keep (N_{Fourier}) when doing field calculations.

Line 2 This tells gives the extent of the field map (from z_{start} to z_{end}) relative to the `ELEMEDGE` of the field map, and how many grid spacings there are in the field map.

Line 3 Field frequency.

Line 4 If one sets `FAST = TRUE` for the field map, this tells OPAL-T the radial extent of the internally generated 2D field map. Otherwise this line is ignored. (Although it must always be present.)

The lines following the header give the 1D field map grid values from 1 to $N_z + 1$. From these, $N_z/2$ complex Fourier coefficients are calculated, of which only $N_{Fourier}$ are used when finding field values during the simulation. Figure C.5 gives an example of a `1DDynamic` field file.

C.10 AstraDynamic

```
AstraDynamic 40
2997.924
0.00000000e+00 0.00000000e+00
5.0007941e-04 2.8090000e-04
9.9991114e-04 5.6553000e-04
1.4996762e-03 8.4103000e-04
. . . .
1.9741957e-01 1.4295000e-03
1.9792448e-01 1.1306000e-03
1.9841987e-01 8.4103000e-04
1.9891525e-01 5.6553000e-04
1.9942016e-01 2.8090000e-04
1.9991554e-01 0.0000000e+00
```

Figure C.6: A 1D field map describing a dynamic field using N non-equidistant grid points in longitudinal direction. From these N non-equidistant field values N equidistant field values are computed from which in turn $N/2$ complex Fourier coefficients are calculated. In this example only 40 Fourier coefficients are kept when calculating field values during the simulation. The z -position of each sampling is in the 1st column (in meters), the corresponding longitudinal on-axis electric field amplitude is in the 2nd column. OPAL-T normalizes the field values such that $\max(|E_{\text{on axis}}|) = 1$ MV/m. The frequency of this field is 2997.924 MHz. (OPAL-T does not provide a FAST version of this map type.)

Table C.4: Layout of an AstraDynamic field map file.

AstraMagnetostatic	N_{Fourier}
Frequency (in MHz)	
z_1 (in meters)	$E_{z,1}$ (MV/m)
z_2 (in meters)	$E_{z,s}$ (MV/m)
.	
.	
.	
z_N (in meters)	$E_{z,N}$ (MV/m)

An AstraDynamic field map has the general form shown in Table C.4. The first line forms the file header and tells OPAL-T how the field map data is being presented:

Line 1 This tells OPAL-T what type of field file it is (AstraDynamic) and how many Fourier coefficients to keep (N_{Fourier}) when doing field calculations.

Line 2 Field frequency.

The lines following the header gives N non-equidistant field values and their corresponding z positions (relative to `ELEMEDGE`). From these, OPAL-T will use cubic spline interpolation to find N equidistant field values within the range defined by the z positions. From these equidistant field values, $N/2$ complex Fourier coefficients are calculated, of which only N_{Fourier} are used when finding field values during the simulation.

Figure C.6 gives an example of an `AstraDynamic` field file.

C.11 1DProfile1

A `1DProfile1` field map is used to define Enge functions [44] that describe the fringe fields for the entrance and exit of a magnet:

$$F(z) = \frac{1}{1 + e^{\sum_{n=0}^{N_{order}} c_n (z/D)^n}}$$

where D is the full gap of the magnet, N_{order} is the Enge function order and z is the distance from the Enge function origin perpendicular to the edge of the magnet. The constants, c_n , and the Enge function origin are fitted parameters chosen to best represent the fringe field of the magnet being modeled.

A `1DProfile1` field map describes two Enge functions: one for the magnet entrance and one for the magnet exit. An illustration of this is shown in Figure C.7. In the top part of the figure we see a plot of the relative magnet field strength along the mid-plane for a rectangular dipole magnet. To describe this field with a `1DProfile1` field map, an Enge function is fit to the entrance fringe field between `zbegin_entry` and `zend_entry` in the figure, using the indicated entrance origin. Likewise, an Enge function is fit to the exit fringe field between `zbegin_exit` and `zend_exit` using the indicated exit origin. The parameters for these two Enge functions are subsequently entered into a `1DProfile1` field map, as described below.

Currently, `1DProfile1` field maps are only implemented for `RBEND` and `SBEND` elements (see 8.4.1 8.4.2, and 8.4.4).

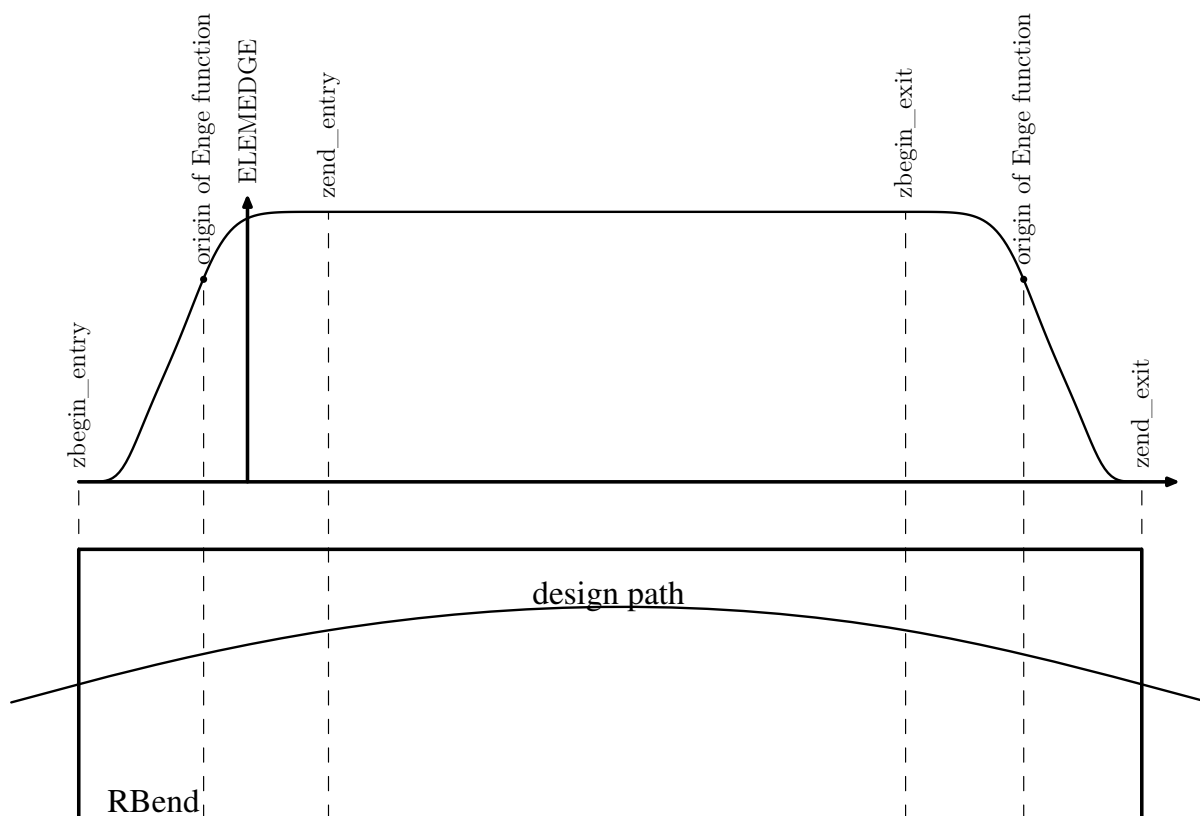


Figure C.7: Example of Engage functions describing the entrance and exit fringe fields of a rectangular bend magnet. The top part of the figure shows the relative field strength on the mid-plane. The bottom part of the figure shows an example of a particle trajectory through the magnet. Note that the magnet field is naturally divided into three regions: entrance fringe field, central field, and exit fringe field.

A 1DProfile1 field map has the general form shown in Table C.5. The first three lines form the file header and tell OPAL-T how the field map data is being presented:

- Line 1** This tells OPAL-T what type of field file it is (1DProfile1), the Enge coefficient order for the entrance fringe fields ($N_{Enge Entrance}$), the Enge coefficient order for the exit fringe fields ($N_{Enge Exit}$), and the gap of the magnet.
- Line 2** The first three values on the second line are used to define the extent of the fringe fields for the entrance region of the magnet. This can be done two different ways as will be described below (see C.11.1 and C.11.2). The fourth value on line 2 is not currently used (but must still be present).
- Line 3** The first three values on the third line are used to define the extent of the fringe fields for the exit region of the magnet. This can be done two different ways as will be described below (see C.11.1 and C.11.2). The fourth value on line 3 is not currently used (but must still be present).

The lines following the three header lines give the entrance region Enge coefficients from c_0 to $c_{N_{Enge Entrance}}$, followed by the exit region Enge coefficients from c_0 to $c_{N_{Enge Exit}}$.

There are two types of 1DProfile1 field map files: 1DProfile Type 1 and 1DProfile Type 2. The difference between the two is a small change in how the entrance and exit fringe field regions are described. This will be explained in C.11.1 (1DProfile Type 1) and C.11.2 (1DProfile Type 2).

Table C.5: Layout of a 1DProfile1 field map file.

1DProfile1	$N_{Enge Entrance}$	$N_{Enge Exit}$	Gap (in cm)
Entrance Parameter 1 (in cm)	Entrance Parameter 2 (in cm)	Entrance Parameter 3	Place Holder
Exit Parameter 1 (in cm)	Exit Parameter 2 (in cm)	Exit Parameter 3	Place Holder
$c_0 Entrance$			
$c_1 Entrance$			
.			
.			
.			
$c_{N_{Enge Entrance}}$			
$c_0 Exit$			
$c_1 Exit$			
.			
.			
.			
$c_{N_{Enge Exit}}$			

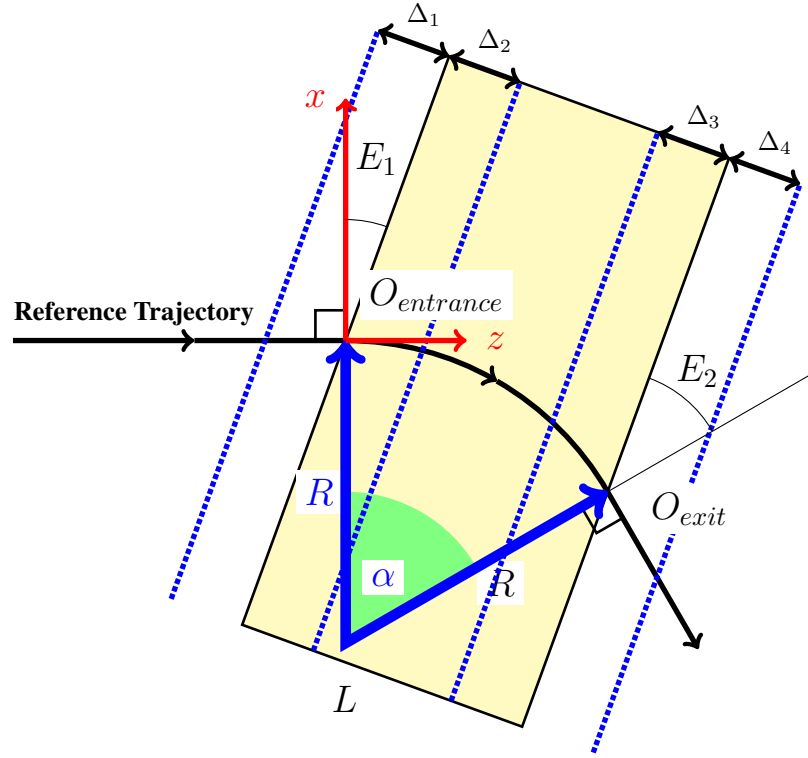


Figure C.8: Illustration of a rectangular bend (RBEND see 8.4.1) showing the entrance and exit fringe field regions. Δ_1 is the perpendicular distance in front of the entrance edge of the magnet where the magnet fringe fields are non-negligible. Δ_2 is the perpendicular distance behind the entrance edge of the magnet where the entrance Enge function stops being used to calculate the magnet field. The reference trajectory entrance point is indicated by $O_{entrance}$. Δ_3 is the perpendicular distance in front of the exit edge of the magnet where the exit Enge function starts being used to calculate the magnet field. (In the region between Δ_2 and Δ_3 the field of the magnet is a constant value.) Δ_4 is the perpendicular distance after the exit edge of the magnet where the magnet fringe fields are non-negligible. The reference trajectory exit point is indicated by O_{exit} .

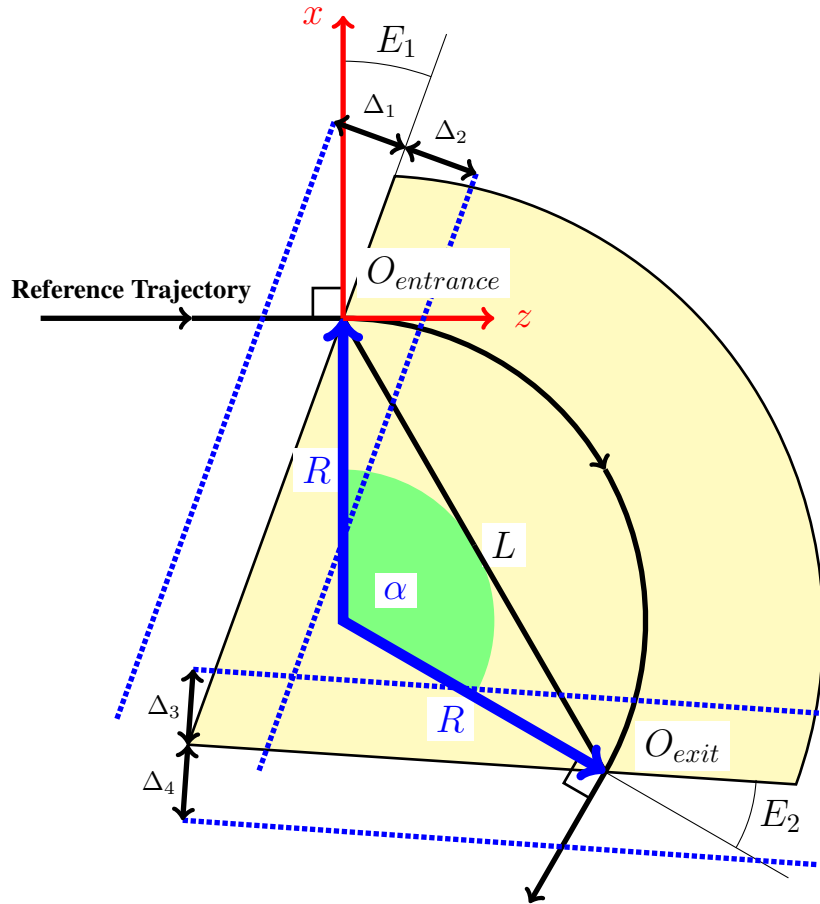


Figure C.9: Illustration of a sector bend (SBEND see 8.4.2) showing the entrance and exit fringe field regions. Δ_1 is the perpendicular distance in front of the entrance edge of the magnet where the magnet fringe fields are non-negligible. Δ_2 is the perpendicular distance behind the entrance edge of the magnet where the entrance Enge function stops being used to calculate the magnet field. The reference trajectory entrance point is indicated by $O_{entrance}$. Δ_3 is the perpendicular distance in front of the exit edge of the magnet where the exit Enge function starts being used to calculate the magnet field. (In the region between Δ_2 and Δ_3 the field of the magnet is a constant value.) Δ_4 is the perpendicular distance after the exit edge of the magnet where the magnet fringe fields are non-negligible. The reference trajectory exit point is indicated by O_{exit} .

C.11.1 1DProfile1 Type 1 for Bend Magnet

A 1DProfile1 Type 1 field map is the same 1DProfile field map found in versions of OPAL previous to OPAL 1.2.0. Figures C.8 and C.9 illustrate the fringe field regions for an RBEND and an SBEND element. Referring to the general field map file shown in Table C.5, the values on lines 2 and 3 are given by:

$$\text{Entrance Parameter 1} = \text{Entrance Parameter 2} - \Delta_1$$

$$\text{Entrance Parameter 3} = \text{Entrance Parameter 2} + \Delta_2$$

$$\text{Exit Parameter 2} = L - \text{Entrance Parameter 2}$$

$$\text{Exit Parameter 1} = \text{Exit Parameter 2} - \Delta_3$$

$$\text{Exit Parameter 3} = \text{Exit Parameter 2} + \Delta_4$$

The value of *Entrance Parameter 2* can be any value. OPAL only cares about the relative differences between parameters. Also note that, internally, the origins of the entrance and exit Enge functions correspond to the reference trajectory entrance and exit points (see Figures C.8 and C.9).

Internally, OPAL reads in a 1DProfile Type 1 map and uses the provided parameters to calculate the values of:

$$L = \text{Exit Parameter 2} - \text{Entrance Parameter 2}$$

$$\Delta_1 = \text{Entrance Parameter 2} - \text{Entrance Parameter 1}$$

$$\Delta_2 = \text{Entrance Parameter 3} - \text{Entrance Parameter 2}$$

$$\Delta_3 = \text{Exit Parameter 2} - \text{Exit Parameter 1}$$

$$\Delta_4 = \text{Exit Parameter 3} - \text{Exit Parameter 2}$$

These values, combined with the entrance fringe field Enge coefficients c_0 through $c_{N_{\text{EngeEntrance}}}$ and exit fringe field Enge coefficients c_0 through $c_{N_{\text{EngeExit}}}$, allow OPAL to find field values anywhere within the magnet. (Again, note that a 1DProfile Type 1 map always places the entrance Enge function origin at the entrance point of the reference trajectory and the exit Enge function origin at the exit point of the reference trajectory.)

Figure C.10 shows an example of a 1DProfile1 Type 1 field map file.

```
1DProfile1 6 7 3.0
-6.0 -2.0 2.0 1000
24.0 28.0 32.0 0
0.00000e+00
4.36222e-06
8.83270e-06
+ 9 lines
1.32490e-05
1.73710e-05
2.18598e-05
```

Figure C.10: A 1D field map describing the fringe field of an element using 7 Enge coefficients for the entrance fringe field and 8 Enge coefficients for the exit fringe field (polynomial order 6 and 7 respectively). The element has a gap height of 3.0 cm, and a length of 30.0 cm. The entrance fringe field is non-negligible from 4.0 cm in front of the magnet's entrance edge and reaches the core strength at 4.0 cm behind the entrance edge of the magnet. (The entrance edge position is given by the element's ELEMEDGE attribute.) The exit fringe field region begins 4.0 cm in front of the exit edge of the magnet and is non-negligible 4.0 cm after the exit edge of the magnet. The value 1000 at the end of line 2 and 0 at the end of line 3 do not have any meaning.

C.11.2 1DProfile1 Type 2 for Bend Magnet

The 1DProfile1 Type 2 field map file format was introduced in OPAL 1.2.0 to allow for more flexibility when defining the Enge functions for the entrance and exit fringe fields. Specifically, a 1DProfile1 Type 2 map does not contain any information about the length of the magnet. Instead, that value is set using the element's L attribute. In turn, this allows us the freedom to make slight changes to how the parameters on lines 2 and 3 of the field map file shown in Table C.5 are defined. Now

Entrance Parameter 2 = \perp distance of entrance Enge function origin from magnet entrance edge

Exit Parameter 2 = \perp distance of exit Enge function origin from magnet exit edge

The other parameters are defined the same as before:

$$\text{Entrance Parameter 1} = \text{Entrance Parameter 2} - \Delta_1$$

$$\text{Entrance Parameter 3} = \text{Entrance Parameter 2} + \Delta_2$$

$$\text{Exit Parameter 1} = \text{Exit Parameter 2} - \Delta_3$$

$$\text{Exit Parameter 3} = \text{Exit Parameter 2} + \Delta_4$$

As before, internally, OPAL reads in a 1DProfile Type 2 map and uses the provided parameters to calculate the values of:

$$\Delta_1 = \text{Entrance Parameter 2} - \text{Entrance Parameter 1}$$

$$\Delta_2 = \text{Entrance Parameter 3} - \text{Entrance Parameter 2}$$

$$\Delta_3 = \text{Exit Parameter 2} - \text{Exit Parameter 1}$$

$$\Delta_4 = \text{Exit Parameter 3} - \text{Exit Parameter 2}$$

These values, combined with the length of the magnet, L (set by the element attribute) and the entrance fringe field Enge coefficients c_0 through $c_{N_{\text{Enge_Entrance}}}$ and exit fringe field Enge coefficients c_0 through $c_{N_{\text{Enge_Exit}}}$, allow OPAL to find field values anywhere within the magnet.

The 1DProfile1 Type 2 field map file format has two main advantages:

1. The Enge function origins can be adjusted to more accurately model a magnet's fringe fields as they are no longer fixed to the entrance and exit points of the reference trajectory.
2. Two magnets with the same fringe fields, but different lengths, can be modeled with a single 1DProfile Type 2 field map file rather than two separate files.

Figure C.11 shows an example of a 1DProfile1 Type 2 field map file.

```

1DProfile1 6 7 3.0
-6.0 -2.0 2.0 0
-2.0 2.0 6.0 0
  0.00000e+00
  4.36222e-06
  8.83270e-06
+ 9 lines
  1.32490e-05
  1.73710e-05
  2.18598e-05

```

Figure C.11: A 1D field map describing the fringe field of an element using 7 Enge coefficients for the entrance fringe field and 8 Enge coefficients for the exit fringe field (polynomial order 6 and 7 respectively). The element has a gap height of 3.0 cm. The entrance fringe field is non-negligible from 4.0 cm in front of the magnet's entrance edge and reaches the core strength at 4.0 cm behind the entrance edge of the magnet. The exit fringe field region begins 4.0 cm in front of the exit edge of the magnet and is non-negligible 4.0 cm after the exit edge of the magnet. The value 0 at the end of line 2 and 0 at the end of line 3 do not have any meaning. The entrance Enge function origin is 2.0 cm in front (upstream) of the magnet's entrance edge. The exit Enge function origin is 2.0 cm behind (downstream of) the exit edge of the magnet.

C.12 2DElectroStatic

```

2DElectroStatic XZ
-3.0 51.0 4999
0.0 2.0 199
  0.000000e+00  0.000000e+00
  4.36222e-06  0.000000e+00
  8.83270e-06  0.000000e+00
+ 999994 lines
  1.32490e-05  0.000000e+00
  1.73710e-05  0.000000e+00
  2.18598e-05  0.000000e+00

```

Figure C.12: A 2D field map describing an electrostatic field using 5000 grid points in the longitudinal direction times 200 grid points in the radial direction. The field between the grid points is calculated using bilinear interpolation. The field is non-negligible from -3.0 cm to 51.0 cm relative to ELEMEDGE and the 200 grid points in the radial direction span the distance from 0.0 cm to 2.0 cm . The field values are ordered in XZ orientation, so the index in the longitudinal direction changes fastest and therefore E_z values are stored in the first column and E_r values in the second (see C.5). OPAL-T normalizes the field so that $\max(|E_{z, \text{on axis}}|) = 1\text{ MV m}^{-1}$.

Table C.6: Layout of a 2DElectroStatic field map file.

2DElectroStatic	Orientation (XZ or ZX)	
z_{start} (or r_{start}) (in cm)	z_{end} (or r_{end}) (in cm)	N_z (or N_r)
r_{start} (or z_{start}) (in cm)	r_{end} (or z_{end}) (in cm)	N_r (or N_z)
$E_{z,1}$ (or $E_{r,1}$) (MV/m)	$E_{r,1}$ (or $E_{z,1}$) (MV/m)	
$E_{z,2}$ (or $E_{r,2}$) (MV/m)	$E_{r,2}$ (or $E_{z,2}$) (MV/m)	
.		
.		
.		
$E_{z,N}$ (or $E_{r,N}$) (MV/m)	$E_{r,N}$ (or $E_{z,N}$) (MV/m)	

A 2DElectroStatic field map has the general form shown in Table C.6. The first three lines form the file header and tell OPAL-T how the field map data is being presented:

- Line 1** This tells OPAL-T what type of field file it is (2DElectroStatic) and the field orientation (see C.5).
- Line 2** This gives the extent of the field map and how many grid spacings there are in the fastest changing index direction (see C.5).
- Line 3** This gives the extent of the field map and how many grid spacings there are in the slowest changing index direction (see C.5).

The lines following the header give the 2D field map grid values from 1 to $N = (N_z + 1) \times (N_r + 1)$. The order of these depend on the field orientation (see C.5) and can be one of two formats:

If Orientation = XZ: E_z (MV/m) E_r (MV/m)

If Orientation = ZX: E_r (MV/m) E_z (MV/m)

Figure C.12 gives an example of a 2DElectroStatic field file.

C.13 2DMagnetoStatic

```

2DMagnetoStatic ZX
0.0 2.0 199
-3.0 51.0 4999
  0.000000e+00  0.000000e+00
  0.000000e+00  4.36222e-06
  0.000000e+00  8.83270e-06
+ 999994 lines
  0.000000e+00  1.32490e-05
  0.000000e+00  1.73710e-05
  0.000000e+00  2.18598e-05

```

Figure C.13: A 2D field map describing a magnetostatic field using 5000 grid points in the longitudinal direction times 200 grid points in the radial direction. The field between the grid points is calculated using bilinear interpolation. The field is non-negligible from -3.0 cm to 51.0 cm relative to ELEMEDGE and the 200 grid points in the radial direction span the distance from 0.0 cm to 2.0 cm . The field values are ordered in the ZX orientation, so the index in the radial direction changes fastest and therefore B_r values are stored in the first column and B_z values in the second (see C.5). OPAL-T normalizes the field so that $\max(|B_{z, \text{ on axis}}|) = 1\text{ T}$.

Table C.7: Layout of a 2DMagnetoStatic field map file.

2DMagnetoStatic	Orientation (XZ or ZX)	
z_{start} (or r_{start}) (in cm)	z_{end} (or r_{end}) (in cm)	N_z (or N_r)
r_{start} (or z_{start}) (in cm)	r_{end} (or z_{end}) (in cm)	N_r (or N_z)
$B_{z,1}$ (or $B_{r,1}$) (T)	$B_{r,1}$ (or $B_{z,1}$) (T)	
$B_{z,2}$ (or $B_{r,2}$) (T)	$B_{r,2}$ (or $B_{z,2}$) (T)	
.		
.		
.		
$B_{z,N}$ (or $B_{r,N}$) (T)	$B_{r,N}$ (or $B_{z,N}$) (T)	

A 2MagnetoStatic field map has the general form shown in Table C.7. The first three lines form the file header and tell OPAL-T how the field map data is being presented:

Line 1 This tells OPAL-T what type of field file it is (2DMagnetoStatic) and the field orientation (see C.5).

Line 2 This gives the extent of the field map and how many grid spacings there are in the fastest changing index direction (see C.5).

Line 3 This gives the extent of the field map and how many grid spacings there are in the slowest changing index direction (see C.5).

The lines following the header give the 2D field map grid values from 1 to $N = (N_z + 1) \times (N_r + 1)$. The order of these depend on the field orientation (see C.5) and can be one of two formats:

If Orientation = XZ: B_z (T) B_r (T)

If Orientation = ZX: B_r (T) B_z (T)

Figure C.13 gives an example of a 2DMagNetoStatic field file.

C.14 2DDynamic

```

2DDynamic XZ
-3.0 51.0 4121
1498.953425154
0.0 1.0 75
  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
  4.36222e-06  0.00000e+00  0.00000e+00  4.36222e-06
  8.83270e-06  0.00000e+00  0.00000e+00  8.83270e-06
+ 313266 lines
  1.32490e-05  0.00000e+00  0.00000e+00  1.32490e-05
  1.73710e-05  0.00000e+00  0.00000e+00  1.73710e-05
  2.18598e-05  0.00000e+00  0.00000e+00  2.18598e-05

```

Figure C.14: A 2D field map describing a dynamic field oscillating with a frequency of 1498.953425154 MHz. The field map provides 4122 grid points in the longitudinal direction times 76 grid points in radial direction. The field between the grid points is calculated with a bilinear interpolation. The field is non-negligible between -3.0 cm and 51.0 cm relative to ELEMEDGE and the 76 grid points in radial direction span the distance from 0.0 cm to 1.0 cm . The field values are ordered in the XZ orientation, so the index in the longitudinal direction changes fastest and therefore E_z values are stored in the first column and E_r values in the second. The third column contains the electric field magnitude, $|E|$, and is not used (but must still be included). The fourth column is H_ϕ in A/m. The third and fourth columns are always the same and do not depend on the field orientation (see C.5). OPAL-T normalizes the field so that $\max(|E_{z, \text{on axis}}|) = 1\text{ MV m}^{-1}$.

Table C.8: Layout of a 2DDynamic field map file.

2DDynamic		Orientation (XZ or ZX)	
z_{start} (or r_{start}) (in cm)	z_{end} (or r_{end}) (in cm)	N_z (or N_r)	
<i>Frequency</i> (in MHz)			
r_{start} (or z_{start}) (in cm)	r_{end} (or z_{end}) (in cm)	N_r (or N_z)	
$E_{z,1}$ (or $E_{r,1}$) (MV/m)	$E_{r,1}$ (or $E_{z,1}$) (MV/m)	$ E_1 $ (MV/m)	$H_{\phi,1}$ (A/m)
$E_{z,2}$ (or $E_{r,2}$) (MV/m)	$E_{r,2}$ (or $E_{z,2}$) (MV/m)	$ E_2 $ (MV/m)	$H_{\phi,2}$ (A/m)
.	.	.	.
.	.	.	.
.	.	.	.
$E_{z,N}$ (or $E_{r,N}$) (MV/m)	$E_{r,N}$ (or $E_{z,N}$) (MV/m)	$ E_N $ (MV/m)	$H_{\phi,N}$ (A/m)

A 2DDynamic field map has the general form shown in Table C.8. The first four lines form the file header and tell OPAL-T how the field map data is being presented:

Line 1 This tells OPAL-T what type of field file it is (2DDynamic) and the field orientation (see C.5).

Line 2 This gives the extent of the field map and how many grid spacings there are in the fastest changing index direction (see C.5).

Line 3 Field frequency.

Line 4 This gives the extent of the field map and how many grid spacings there are in the slowest changing index direction (see C.5).

The lines following the header give the 2D field map grid values from 1 to $N = (N_z + 1) \times (N_r + 1)$. The order of these depend on the field orientation (see C.5) and can be one of two formats:

If Orientation = XZ: E_z (MV/m) E_r (MV/m) $|E|$ (MV/m) H_ϕ (A/m)

If Orientation = ZX: E_r (MV/m) E_z (MV/m) $|E|$ (MV/m) H_ϕ (A/m)

The third item (the field magnitude) on each data line is not used by OPAL-T but must be there.

Figure C.14 gives an example of a 2DDynamic field file.

C.15 3DDynamic

```

3DDynamic XYZ
1498.953425154
-1.5 1.5 227
-1.0 1.0 151
-3.0 51.0 4121
0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
4.36e-06 0.00e+00 4.36e-06 0.00e+00 4.36e-06 0.00e+00
8.83e-06 0.00e+00 8.83e-06 0.00e+00 8.83e-06 0.00e+00
+ 142'852'026 lines
1.32e-05 0.00e+00 1.32e-05 0.00e+00 1.32e-05 0.00e+00
1.73e-05 0.00e+00 1.73e-05 0.00e+00 1.73e-05 0.00e+00
2.18e-05 0.00e+00 2.18e-05 0.00e+00 2.18e-05 0.00e+00

```

Figure C.15: A 3D field map describing a dynamic field oscillating with $1.498953425154\text{ GHz}$. The field map provides 4122 grid points in z-direction times 228 grid points in x-direction and 152 grid points in y-direction. The field between the grid points is calculated with a bilinear interpolation. The field is non-negligible between -3.0 cm to 51.0 cm relative to ELEMEDGE, the 228 grid points in x-direction range from -1.5 cm to 1.5 cm and the 152 grid points in y-direction range from -1.0 cm to 1.0 cm relative to the design path. The field values are ordered in XYZ orientation, the index in z-direction changes fastest, then the index in y-direction while the index in x-direction changes slowest. This is the only orientation that is implemented. The columns correspond to E_x , E_y , E_z , H_x , H_y and H_z .

Table C.9: Layout of a 3DDynamic field map file.

3DDynamic	Orientation (XYZ)				
<i>Frequency</i> (in MHz)					
x_{start} (in cm)	x_{end} (in cm)	N_x			
y_{start} (in cm)	y_{end} (in cm)	N_y			
z_{start} (in cm)	z_{end} (in cm)	N_z			
$E_{x,1}$ (MV/m))	$E_{y,1}$ (MV/m)	$E_{z,1}$ (MV/m)	$H_{x,1}$ (A/m)	$H_{y,1}$ (A/m)	$H_{z,1}$ (A/m)
$E_{x,2}$ (MV/m))	$E_{y,2}$ (MV/m)	$E_{z,2}$ (MV/m)	$H_{x,2}$ (A/m)	$H_{y,2}$ (A/m)	$H_{z,2}$ (A/m)
.					
.					
.					
$E_{x,N}$ (MV/m))	$E_{y,N}$ (MV/m)	$E_{z,N}$ (MV/m)	$H_{x,N}$ (A/m)	$H_{y,N}$ (A/m)	$H_{z,N}$ (A/m)

A 3DDynamic field map has the general form shown in Table C.9. The first five lines form the file header and

tell OPAL-T how the field map data is being presented:

Line 1 This tells OPAL-T what type of field file it is (`3DDynamic`) and the field orientation (see C.5). Currently only an XYZ orientation is implemented.

Line 2 Field frequency.

Line 3 This gives the extent of the field map and how many grid spacings there are in the fastest changing index direction (see C.5).

Line 4 This gives the extent of the field map and how many grid spacings there are in the next fastest changing index direction (see C.5).

Line 5 This gives the extent of the field map and how many grid spacings there are in the slowest changing index direction (see C.5).

The lines following the header give the 3D field map grid values from 1 to $N = (N_z + 1) \times (N_y + 1) \times (N_x + 1)$. The order of these depend on the field orientation (see C.5) and can currently only be the format shown in Table C.9. Figure C.15 gives an example of a `3DDynamic` field file.

Appendix D

OPAL - MADX Conversion Guide

We note with α, β and γ the Twiss parameters.

$$\begin{aligned}\sigma_{beam} &= \begin{pmatrix} \sigma_x & \sigma_{xp_x} \\ \sigma_{xp_x} & \sigma_{p_x} \end{pmatrix} = \begin{pmatrix} \sigma_x & \delta \cdot \sqrt{\sigma_x \sigma_{p_x}} \\ \delta \cdot \sqrt{\sigma_x \sigma_{p_x}} & \sigma_{p_x} \end{pmatrix} = \begin{pmatrix} \langle x^2 \rangle & \langle xp_x \rangle \\ \langle xp_x \rangle & \langle p_x^2 \rangle \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{N} \sum_{i=1}^N x_i^2 & \frac{1}{N} \sum_{i=1}^N x_i p_{x_i} \\ \frac{1}{N} \sum_{i=1}^N x_i p_{x_i} & \frac{1}{N} \sum_{i=1}^N p_{x_i}^2 \end{pmatrix} = \varepsilon \cdot \begin{pmatrix} \beta & -\alpha \\ -\alpha & \gamma \end{pmatrix}\end{aligned}$$

$$\begin{aligned}\bar{p}_x &= \sqrt{\frac{1}{N} \sum_{i=1}^N p_{x_i}^2} &= \sqrt{\sigma_{p_x}} &\bar{x} &= \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2} \\ \bar{p}_y &= \sqrt{\frac{1}{N} \sum_{i=1}^N p_{y_i}^2} &= \sqrt{\sigma_{p_y}} &\bar{y} &= \sqrt{\frac{1}{N} \sum_{i=1}^N y_i^2}\end{aligned}$$

$$\begin{aligned}\gamma &= \frac{E_{kin} + m_p}{m_p} &\beta &= \sqrt{1 - \frac{1}{\gamma^2}} = \frac{v}{c} \\ (\beta\gamma) &= \frac{E_{kin} + m_p}{m_p} \cdot \sqrt{1 - \frac{1}{\gamma^2}} = \frac{\beta}{\sqrt{1 - \beta^2}} &\mathbf{B}\rho &= \frac{(\beta\gamma) \cdot m_p \cdot 10^9}{c} \text{ [T m]} \\ m_p &= 0.939277 \text{ [GeV]} &c &= 299792458 \text{ [m/s]}\end{aligned}$$

Quantity	MADX	Conversion	OPAL-Output
Momenta	\bar{p}_x [rad]	$\bar{p}_x [\beta\gamma] = (\bar{p}_x [\text{rad}]) \cdot (\beta\gamma)$	\bar{p}_x [$\beta\gamma$]
Correlation of \bar{x}, \bar{p}_x	δ [1]	$= (\sigma_{xp_x} [\text{m rad}] / ((\bar{p}_x [\text{rad}]) \cdot (\bar{x} [\text{m}]))$ $= (\sigma_{xp_x} [\text{m rad}] / \sqrt{(\sigma_x [\text{m}^2]) \cdot (\sigma_{p_x} [\text{rad}^2])})$	δ [1]
Emitance	ε_x [m rad]	$\varepsilon_x [\text{m } \beta\gamma] = \sqrt{(\bar{p}_x [\beta\gamma])^2 \cdot (\bar{x} [\text{m}])^2 - (\delta \cdot (\bar{x} [\text{m}]) \cdot (\bar{p}_x [\beta\gamma]))^2}$ $= \sqrt{(\sigma_{p_x} [(\beta\gamma)^2]) \cdot (\sigma_x [\text{m}^2]) - \left(\delta \cdot \sqrt{(\sigma_x [\text{m}^2]) \cdot (\sigma_{p_x} [(\beta\gamma)^2])} \right)^2}$ $= \sqrt{(\sigma_{p_x} [(\beta\gamma)^2]) \cdot (\sigma_x [\text{m}^2]) - (\sigma_{xp_x} [\text{m } \beta\gamma])^2}$	ε_x [m $\beta\gamma$]
Twiss Parameter α	α [1]	$\alpha [1] = -\delta \cdot (\bar{x} [\text{m}]) \cdot (\bar{p}_x [\beta\gamma]) / (\varepsilon_x [\text{m } \beta\gamma])$ $= -\delta \cdot \sqrt{(\sigma_x [\text{m}^2]) \cdot (\sigma_{p_x} [(\beta\gamma)^2])} / (\varepsilon_x [\text{m } \beta\gamma])$	α_T [1]
Twiss Parameter β_T	β_T [m/rad]	$\beta_T [\text{m}/\beta\gamma] = (\bar{x} [\text{m}])^2 / (\varepsilon_x [\text{m } \beta\gamma])$ $= (\sigma_x [\text{m}^2]) / (\varepsilon_x [\text{m } \beta\gamma])$	β_T [m/ $\beta\gamma$]
Twiss Parameter γ_T	γ_T [rad/m]	$\gamma_T [\beta\gamma/\text{m}] = (\bar{p}_x [\beta\gamma])^2 / (\varepsilon_x [\text{m } \beta\gamma])$ $= (\sigma_{p_x} [(\beta\gamma)^2]) / (\varepsilon_x [\text{m } \beta\gamma])$	γ_T [$\beta\gamma/\text{m}$]
Focusing strength	k_1 [m ⁻²]	$k_1 [\text{T}/\text{m}] = (k_1 [\text{m}^{-2}]) \cdot (\text{B}_\rho [\text{T m}])$	k_1 [T/m]
Quantity	MADX	Conversion	OPAL-Input
Element Position	at := [m] Center of the element	ELEMEDGE = (Center of the element) - (Length of the element)/2	ELEMEDGE = [m] Begin of the element
Quantity	OPAL-Output	Conversion	OPAL-Input
Momenta	\bar{p}_x [$\beta\gamma$]	$p_x [\text{eV}] = m_p \cdot 10^9 \cdot \left(\sqrt{(\bar{p}_x [\beta\gamma])^2 + 1} - 1 \right)$	\bar{p}_x [eV]

Appendix E

Autophasing Algorithm

E.1 Standing Wave Cavity

In OPAL-T the elements are implemented as external fields that are read in from a file. The fields are described by a 1D, 2D or 3D sampling (equidistant or non-equidistant). To get the actual field at any position a linear interpolation multiplied by $\cos(\omega t + \varphi)$, where ω is the frequency and φ is the lag. The energy gain of a particle then is

$$\Delta E(\varphi, r) = q V_0 \int_{z_{\text{begin}}}^{z_{\text{end}}} \cos(\omega t(z, \varphi) + \varphi) E_z(z, r) dz. \quad (\text{E.1})$$

To maximize the energy gain we have to take the derivative with respect to the lag, φ and set the result to zero:

$$\begin{aligned} \frac{d \Delta E(\varphi, r)}{d \varphi} &= - \int_{z_{\text{begin}}}^{z_{\text{end}}} \left(1 + \omega \frac{\partial t(z, \varphi)}{\partial \varphi}\right) \sin(\omega t(z, \varphi) + \varphi) E_z(z, r) \\ &= - \cos(\varphi) \int_{z_{\text{begin}}}^{z_{\text{end}}} \left(1 + \omega \frac{\partial t(z, \varphi)}{\partial \varphi}\right) \sin(\omega t(z, \varphi)) E_z(z, r) dz \\ &\quad - \sin(\varphi) \int_{z_{\text{begin}}}^{z_{\text{end}}} \left(1 + \omega \frac{\partial t(z, \varphi)}{\partial \varphi}\right) \cos(\omega t(z, \varphi)) E_z(z, r) dz \stackrel{!}{=} 0. \end{aligned} \quad (\text{E.2})$$

Thus to get the maximum energy the lag has to fulfill

$$\tan(\varphi) = -\frac{\Gamma_1}{\Gamma_2}, \quad (\text{E.3})$$

where

$$\Gamma_1 = \sum_{i=1}^{N-1} \left(1 + \omega \frac{\partial t}{\partial \varphi}\right) \int_{z_{i-1}}^{z_i} \sin\left(\omega\left(t_{i-1} + \Delta t_i \frac{z - z_{i-1}}{\Delta z_i}\right)\right) \left(E_{z,i-1} + \Delta E_{z,i} \frac{z - z_{i-1}}{\Delta z_i}\right) dz \quad (\text{E.4})$$

and

$$\Gamma_2 = \sum_{i=1}^{N-1} \left(1 + \omega \frac{\partial t}{\partial \varphi}\right) \int_{z_{i-1}}^{z_i} \cos\left(\omega\left(t_{i-1} + \Delta t_i \frac{z - z_{i-1}}{\Delta z_i}\right)\right) \left(E_{z,i-1} + \Delta E_{z,i} \frac{z - z_{i-1}}{\Delta z_i}\right) dz. \quad (\text{E.5})$$

Between two sampling points we assume a linear correlation between the electric field and position respectively between time and position. The products in the integrals between two sampling points can be expanded and solved analytically. We then find

$$\Gamma_1 = \sum_{i=1}^{N-1} \left(1 + \omega \frac{\partial t}{\partial \varphi}\right) \Delta z_i (E_{z,i-1} (\Gamma_{11,i} - \Gamma_{12,i}) + E_{z,i} \Gamma_{12,i})$$

and

$$\Gamma_1 = \sum_{i=1}^{N-1} \left(1 + \omega \frac{\partial t}{\partial \varphi}\right) \Delta z_i (E_{z,i-1} (\Gamma_{21,i} - \Gamma_{22,i}) + E_{z,i} \Gamma_{22,i})$$

where

$$\begin{aligned} \Gamma_{11,i} &= \int_0^1 \sin(\omega(t_{i-1} + \tau \Delta t_i)) d\tau = -\frac{\cos(\omega t_i) - \cos(\omega t_{i-1})}{\omega \Delta t_i} \\ \Gamma_{12,i} &= \int_0^1 \sin(\omega(t_{i-1} + \tau \Delta t_i)) \tau d\tau = \frac{-\omega \Delta t_i \cos(\omega t_i) + \sin(\omega t_i) - \sin(\omega t_{i-1})}{\omega^2 (\Delta t_i)^2} \\ \Gamma_{21,i} &= \int_0^1 \cos(\omega(t_{i-1} + \tau \Delta t_i)) d\tau = \frac{\sin(\omega t_i) - \sin(\omega t_{i-1})}{\omega \Delta t_i} \\ \Gamma_{22,i} &= \int_0^1 \cos(\omega(t_{i-1} + \tau \Delta t_i)) \tau d\tau = \frac{\omega \Delta t_i \sin(\omega t_i) + \cos(\omega t_i) - \cos(\omega t_{i-1})}{\omega^2 (\Delta t_i)^2} \end{aligned}$$

It remains to find the progress of time with respect to the position. In OPAL this is done iteratively starting with

```
K[i] = K[i-1] + (z[i] - z[0]) * q * V;
b[i] = sqrt(1. - 1. / ((K[i] - K[i-1]) / (2.*m*c^2) + 1)^2);
t[i] = t[0] + (z[i] - z[0]) / (c * b[i])
```

By doing so we assume that the kinetic energy, K , increases linearly and proportional to the maximal voltage. With this model for the progress of time we can calculate φ according to Equation E.3. Next a better model for the kinetic Energy can be calculated using

$$K[i] = K[i-1] + q \Delta z[i] (\cos(\varphi) (E_z[i-1] (\Gamma_{21}[i] - \Gamma_{22}[i]) + E_z[i] \Gamma_{22}[i]) - \sin(\varphi) (E_z[i-1] (\Gamma_{11}[i] - \Gamma_{12}[i]) + E_z[i] \Gamma_{12}[i])).$$

With the updated kinetic energy the time model and finally a new φ , that comes closer to the actual maximal kinetic energy, can be obtained. One can iterate a few times through this cycle until the value of φ has converged.

E.2 Traveling Wave Structure

Auto phasing in a traveling wave structure is just slightly more complicated. The field of this element is composed of a standing wave entry and exit fringe field and two standing waves in between, see Figure E.1.

$$\begin{aligned} \Delta E(\varphi, r) &= q V_0 \int_{z_{\text{begin}}}^{z_{\text{beginCore}}} \cos(\omega t(z, \varphi) + \varphi) E_z(z, r) dz \\ &\quad + q V_{\text{core}} \int_{z_{\text{beginCore}}}^{z_{\text{endCore}}} \cos(\omega t(z, \varphi) + \varphi_{c1} + \varphi) E_z(z, r) dz \\ &\quad + q V_{\text{core}} \int_{z_{\text{beginCore}}}^{z_{\text{endCore}}} \cos(\omega t(z, \varphi) + \varphi_{c2} + \varphi) E_z(z + s, r) dz \\ &\quad + q V_0 \int_{z_{\text{endCore}}}^{z_{\text{end}}} \cos(\omega t(z, \varphi) + \varphi_{ef} + \varphi) E_z(z, r) dz, \quad (\text{E.6}) \end{aligned}$$

where s is the cell length. Instead of one sum as in Equation (E.4) and Equation (E.5) there are four sums with different numbers of summands.

Exam FINLB02_RAC: TravelingWave, L=2.80, VOLT=14.750*30/31, NUMCELLS=40,
FMAPFN="FINLB02-RAC.T7", ELEMEDGE=2.67066, MODE=1/3,
FREQ=1498.956, LAG=FINLB02_RAC_lag;

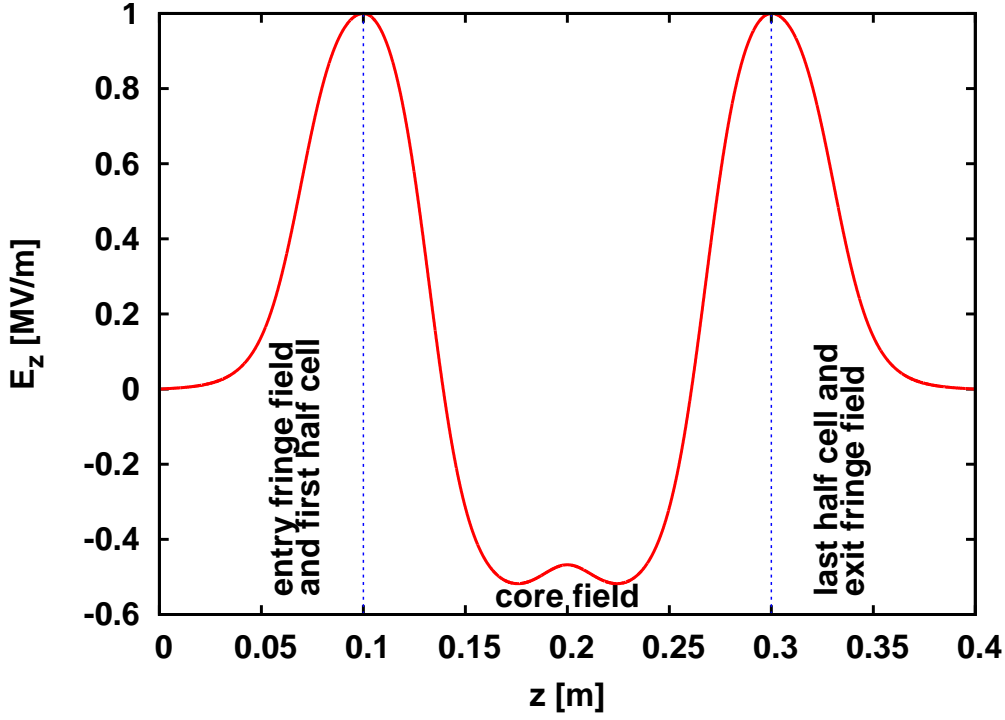


Figure E.1: Field map 'FINLB02-RAC.T7' of type 1DDynamic

For this example we find

$$\begin{aligned}
 V_{\text{core}} &= \frac{V_0}{\sin(2.0/3.0\pi)} = \frac{2V_0}{\sqrt{3.0}} \\
 \varphi_{c1} &= \frac{\pi}{6} \\
 \varphi_{c2} &= \frac{\pi}{2} \\
 \varphi_{\text{ef}} &= -2\pi \cdot (\text{NUMCELLS} - 1) \cdot \text{MODE} = 26\pi
 \end{aligned}$$

E.2.1 Alternative Approache for Traveling Wave Structures

If β doesn't change much along the traveling wave structure (ultra relativistic case) then $t(z, \varphi)$ can be approximated by $t(z, \varphi) = \frac{\omega}{\beta c}z + t_0$. For the example from above the energy gain is approximately

$$\begin{aligned}
 \Delta E(\varphi, r) &= q V_0 \int_0^{1.5 \cdot s} \cos\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \varphi\right) E_z(z, r) dz \\
 &+ \frac{2q V_0}{\sqrt{3}} \int_{1.5 \cdot s}^{40.5 \cdot s} \cos\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \frac{\pi}{6} + \varphi\right) E_z(z, r) dz \\
 &+ \frac{2q V_0}{\sqrt{3}} \int_{1.5 \cdot s}^{40.5 \cdot s} \cos\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \frac{\pi}{2} + \varphi\right) E_z(z + s, r) dz \\
 &+ q V_0 \int_{40.5 \cdot s}^{42 \cdot s} \cos\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \varphi\right) E_z(z, r) dz.
 \end{aligned}$$

Here $\beta c = 2.9886774 \cdot 10^8 \text{ m s}^{-2}$, $\omega = 2\pi \cdot 1.4989534 \cdot 10^9 \text{ Hz}$ and, the cell length, $s = 0.06\bar{6} \text{ m}$. To maximize this energy we have to take the derivative with respect to φ and set the result to 0. We split the field up into the core

field, $E_z^{(1)}$ and the fringe fields (entry fringe field plus first half cell concatenated with the exit fringe field plus last half cell), $E_z^{(2)}$. The core fringe field is periodic with a period of $3s$. We thus find

$$\begin{aligned} 0 \stackrel{!}{=} & \int_0^{1.5s} \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \varphi\right) E_z^{(2)}(z, r) dz \\ & + \frac{2}{\sqrt{3}} \int_0^{39s} \sin\left(\omega\left(\frac{z + 1.5s}{\beta c} + t_0\right) + \frac{\pi}{6} + \varphi\right) E_z^{(1)}(z \bmod(3s), r) dz \\ & + \frac{2}{\sqrt{3}} \int_0^{39s} \sin\left(\omega\left(\frac{z + 1.5s}{\beta c} + t_0\right) + \frac{\pi}{2} + \varphi\right) E_z^{(1)}((z + s) \bmod(3s), r) dz \\ & + \int_{1.5s}^{3s} \sin\left(\omega\left(\frac{z + 39s}{\beta c} + t_0\right) + \varphi\right) E_z^{(2)}(z, r) dz \end{aligned}$$

This equation is much simplified if we take into account that $\omega/\beta c \approx 10\pi$. We then get

$$\begin{aligned} 0 \stackrel{!}{=} & \int_0^{3s} \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \varphi\right) E_z^{(2)}(z) dz \\ & + \frac{26}{\sqrt{3}} \int_0^{3s} \left(\sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \frac{7\pi}{6} + \varphi\right) + \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \frac{5\pi}{6} + \varphi\right) \right) E_z^{(1)}(z) dz \\ & = \int_0^{3s} \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \varphi\right) (E_z^{(2)} - 26 \cdot E_z^{(1)})(z) dz \end{aligned}$$

where we used

$$\begin{aligned} & \int_0^{3s} \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \frac{3\pi}{2} + \varphi\right) E_z^{(1)}((z + s) \bmod(3s), r) dz \\ & \xrightarrow{z'=z+s} \int_s^{4s} \sin\left(\omega\left(\frac{z' - s}{\beta c} + t_0\right) + \frac{3\pi}{2} + \varphi\right) E_z^{(1)}(z' \bmod(3s), r) dz' \\ & = \int_0^{3s} \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \frac{5\pi}{6} + \varphi\right) E_z^{(1)}(z, r) dz. \end{aligned}$$

In the last equal sign we used the fact that both functions, $\sin(\frac{\omega}{\beta c} z)$ and $E_z^{(1)}$ have a periodicity of $3 \cdot s$ to shift the boundaries of the integral.

Using the convolution theorem we find

$$0 \stackrel{!}{=} \int_0^{3s} g(\xi - z)(G - 26 \cdot H)(z) dz = \mathcal{F}^{-1}(\mathcal{F}(g) \cdot (\mathcal{F}(G) - 26 \cdot \mathcal{F}(H)))$$

where

$$\begin{aligned} g(z) &= \begin{cases} -\sin\left(\omega\left(\frac{z}{\beta c} + t_0\right)\right) & 0 \leq z \leq 3 \cdot s \\ 0 & \text{otherwise} \end{cases} \\ G(z) &= \begin{cases} E_z^{(2)}(z) & 0 \leq z \leq 3 \cdot s \\ 0 & \text{otherwise} \end{cases} \\ H(z) &= \begin{cases} E_z^{(1)}(z) & 0 \leq z \leq 3 \cdot s \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

and

$$-\frac{\omega}{\beta c} \xi = \varphi.$$

Here we also used some trigonometric identities:

$$\begin{aligned}
 & \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \pi + \frac{\pi}{6} + \varphi\right) + \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \pi - \frac{\pi}{6} + \varphi\right) \\
 &= -\left(\sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \frac{\pi}{6} + \varphi\right) + \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) - \frac{\pi}{6} + \varphi\right)\right) \\
 &= -2 \cdot \cos\left(\frac{\pi}{6}\right) \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \varphi\right) \\
 &= -\sqrt{3} \sin\left(\omega\left(\frac{z}{\beta c} + t_0\right) + \varphi\right)
 \end{aligned}$$

Appendix F

Benchmarks

F.1 OPAL-T compared with TRANSPORT & TRACE 3D

F.1.1 TRACE 3D

TRACE 3D is an interactive beam dynamics program that calculates the envelopes of a bunched beam, including linear space-charge forces [68]. It provides an instantaneous beam profile diagram and delineates the transverse and longitudinal phase plane, where the ellipses are characterised by the Twiss parameters and emittances (total and unnormalised).

F.1.2 TRACE 3D Units

TRACE 3D supports the following internal coordinates and units for the three phase planes:

- **horizontal plane:**
 - x [mm] is the displacement from the center of the beam bunch;
 - x' [mrad] is the beam divergence;
- **vertical plane:**
 - y [mm] is the displacement from the center of the beam bunch;
 - y' [mrad] is the beam divergence;
- **longitudinal plane:**
 - z [mm] is the displacement from the center of the beam bunch;
 - $\Delta p/p$ [mrad] is the difference between the particle's longitudinal momentum and the reference momentum of the beam bunch.

For input and output, however, z and $\Delta p/p$ are replaced by $\Delta\phi$ [degree] and ΔW [keV], respectively the displacement in phase and energy. The relationships between these longitudinal coordinates are:

$$z = -\frac{\beta\lambda}{360}\Delta\phi \quad (\text{F.1})$$

and

$$\frac{\Delta p}{p} = \frac{\gamma}{\gamma + 1} \frac{\Delta W}{W} \quad (\text{F.2})$$

where β and γ are the relativist parameters, λ is the free-space wavelength of the RF and W is the kinetic energy [MeV] at the beam center. This internal conversion can be displayed using the *command* W (see [68] page 42).

In TRACE 3D, the input beam is described by the following set of parameters:

- **ER**: particle rest mass [MeV/ c^2];
- **Q**: charge state (+1 for protons);
- **W**: beam kinetic energy [MeV]
- **XI**: beam current [mA]
- **BEAMI**: array with initial Twiss parameters in the three phase planes

The alphas are dimensionless, β_x and β_y are expressed in m/rad (or mm/mrad) and β_ϕ in deg/keV;

- $$\text{EMITI} = \varepsilon_x, \varepsilon_y, \varepsilon_\phi$$

The transversal emittances are expressed in π -mm-mrad and in π -deg-keV the longitudinal emittance.

In this beam dynamics code, the total emittance in each phase plane is five times the RMS emittance in that plane and the displayed beam envelopes are $\sqrt{5}$ -times their respective RMS values.

TRACE 3D Graphic Interface

An example of TRACE 3D graphic interface is shown in Figure F.1.

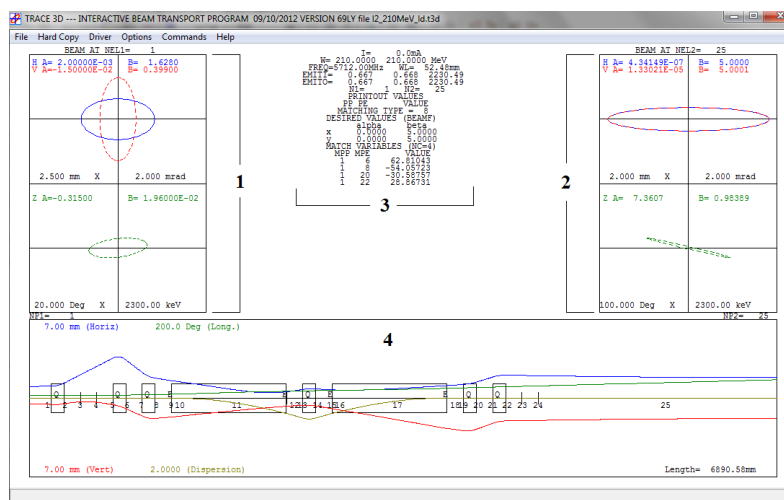


Figure F.1: TRACE 3D graphic interface where: (1) input beam in transverse plane (above) and longitudinal plane (below); (2) output beam in transverse plane (above) and longitudinal plane (below); (3) summary of beam parameters such as input and output emittances and desired value for matching function; (4) line lattice with different elements and beam envelope. The color legend is: blu line for horizontal plane, red line for vertical plane, green line for longitudinal plane and yellow line for dispersion.

F.1.4 TRANSPORT

TRANSPORT is a computer program for first-order and second-order matrix multiplication, intended for the design of beam transport system [9]. The TRANSPORT version for Windows provides a graphic beam profile diagram, as well as a sigma matrix description of the simulated beam and line [69]. Differently from TRACE 3D, the ellipses are characterised by the sigma-matrix coefficients and the Twiss parameters and emittances (total and unnormalised) are reported as output information.

F.1.5 TRANSPORT Units

At any specified position in the system, an arbitrary charged particle is represented by a vector, whose components are positions, angles and momentum of the particle with respect to the reference trajectory. The standard units and internal coordinates in TRANSPORT are:

- **horizontal plane:**
 x [cm] is the displacement of the arbitrary ray with respect to the assumed central trajectory;
 θ [mrad] is the angle the ray makes with respect to the assumed central trajectory;
- **vertical plane:**
 y [cm] is the displacement of the arbitrary ray with respect to the assumed central trajectory;
 ϕ [mrad] is the angle the ray makes with respect to the assumed central trajectory;
- **longitudinal plane:**
 l [cm] is the path length difference between the arbitrary ray and the central trajectory;
 δ [%] is the fractional momentum deviation of the ray from the assumed central trajectory.

Even if TRANSPORT supports this standard set of units [cm, mrad and %]; however using **card 15**, the users can redefine the units (see page 99 on TRANSPORT documentation [9] for more details).

F.1.6 TRANSPORT Input beam

The input beam is described in **card 1** in terms of the semi-axes of a six-dimensional erect ellipsoid beam. In terms of diagonal sigma-matrix elements, the input beam in TRANSPORT is expressed by 7 parameters:

- $\sqrt{\sigma_{ii}}$ [cm] represents one-half of the horizontal ($i=1$), vertical ($i=3$) and longitudinal extent ($i=5$);
- $\sqrt{\sigma_{ii}}$ [mrad] represents one-half of the horizontal ($i=2$), vertical ($i=4$) beam divergence;
- $\sqrt{\sigma_{66}}$ [%] represents one-half of the momentum spread;
- $p(0)$ is the momentum of the central trajectory [GeV/c].

If the input beam is tilted (Twiss alphas not zero), **card 12** must be used, inserting the 15 correlations r_{ij} parameters among the 6 beam components. The correlation parameters are defined as following:

$$r_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \quad (\text{F.3})$$

As explained before, with the **card 15**, it is possible to transform the TRANSPORT standard units in TRACE-like units. In this way, the TRACE 3D sigma-matrix for the input beam, printed out by *command Z*, can be directly used as input beam in TRANSPORT. An example of TRACE 3D sigma-matrix structure is shown in Figure F.2. From the sigma-matrix coefficients, TRANSPORT reports in output the Twiss parameters and the total, unnormalised emittance. Even in this case, a factor 5 is present between the emittances calculated by TRANSPORT and the corresponding RMS values.

x_{\max}					
x'_{\max}	Γ_{12}				
y_{\max}	Γ_{13}	Γ_{23}			
y'_{\max}	Γ_{14}	Γ_{24}	Γ_{34}		
z_{\max}	Γ_{15}	Γ_{25}	Γ_{35}	Γ_{45}	
$\Delta p/p_{\max}$	Γ_{16}	Γ_{26}	Γ_{36}	Γ_{46}	Γ_{56}

Figure F.2: Sigma-matrix structure in TRACE 3D [68]

TRANSPORT Graphic Interface

An improved version of TRANSPORT has been embedded in a new graphic shell written in C++ and is providing GUI type tools, which makes it easier to design new beam lines. A screen shot of a modern GUI Transport interface [69] is shown in Figure F.3.

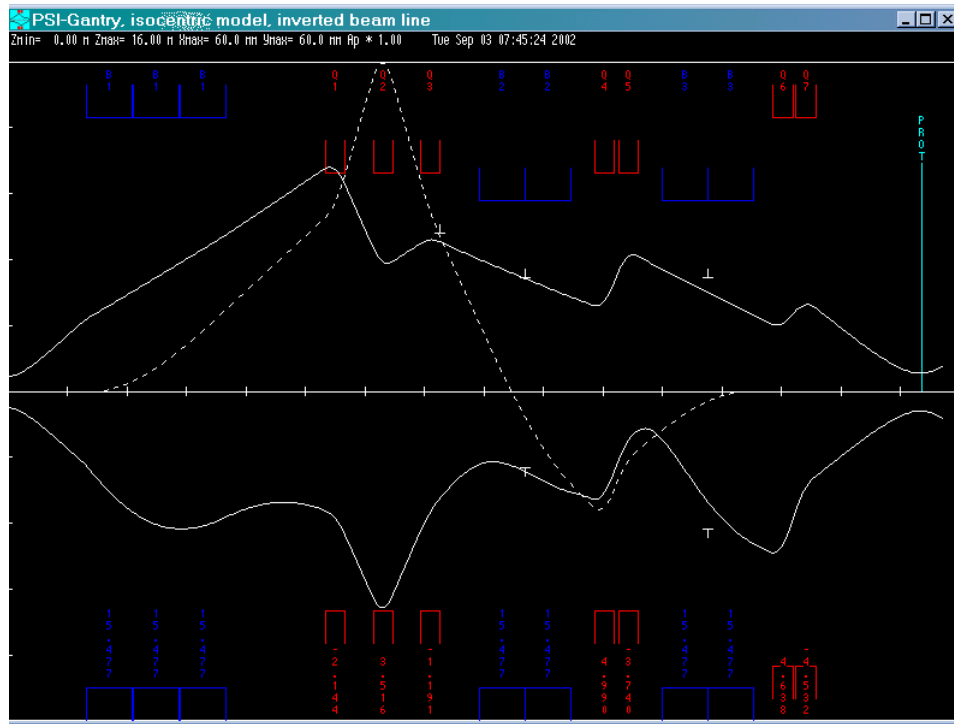


Figure F.3: GUI TRANSPORT graphic interface [70]. The continuous lines describe the beam envelope in the vertical plane (above) and horizontal plane (below). The dashed line displays the dispersion. The elements in the beam line are drawn as blue and red rectangles

F.1.7 Comparison TRACE 3D and TRANSPORT

This study has been done following the same trend of the Regression Test in OPAL [71], replacing the electron beam with a same energy proton beam. Due to the different beam rigidity, the bending magnet features have been redefined with a new magnetic field.

The simulated beam transport line contains:

- drift space (DRIFT 1): 0.250 m length;
- bending magnet (SBEND or RBEND): 0.250 m radius of curvature;
- drift space (DRIFT 2): 0.250 m length.

Keeping fixed the lattice structure, many similar transport lines have been tested adding entrance and exit edge angles to the bending magnet, changing the bending plane (vertical bending magnet) and direction (right or left). In all the cases, the difficulties arise from the non-achromaticity of the system and an increase in the horizontal and longitudinal emittance is expected. In addition, the coupling between these two planes has to be accurately studied.

In the following paragraph, an example of Sector Bending magnet (SBEND) simulation with entrance and exit edge angles is discussed.

Input beam

The starting simulation has been performed with TRACE 3D code. According to section F.1.3, the simulated input beam is described by the following parameters:

```
ER = 938.27
W = 7
FREQ = 700
BEAMI = 0.0, 4.0, 0.0, 4.0, 0.0, 0.0756
EMITI = 0.730, 0.730, 7.56
```

Thanks to the TRACE 3D graphic interface, the input beam can immediately be visualised in the three phase plane as shown in Figure F.4.

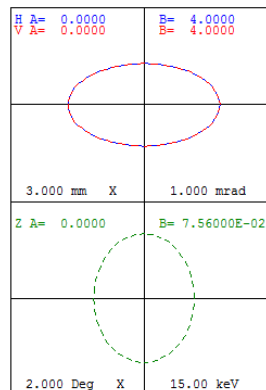


Figure F.4: TRACE 3D input beam in the transversal plane (above) and in the longitudinal plane (below)

The corresponding sigma-matrix with the relative units is displayed by command Z: Before entering the

```
MODIFIED BEAM MATRIX
(mm, mrad, %dp/p)
1.7088
0.4272 0.000
1.7088 0.000 0.000
0.4272 0.000 0.000 0.000
0.1092 0.000 0.000 0.000 0.000
0.0717 0.000 0.000 0.000 0.000 0.000
```

Figure F.5: TRACE 3D sigma-matrix for the input beam

TRACE 3D sigma-matrix coefficients in TRANSPORT, a changing in the units is required using the **card 15** in the following way:

```
15. 1. 'MM' 0.1 ; //express in mm the horizontal and vertical beam size
15. 5. 'MM' 0.1 ; //express in mm the beam length
```

At this point, the TRANSPORT input beam is defined by **card 1**:

```
1.0 1.709 0.427 1.709 0.427 0.11 0.0717 0.1148 /BEAM/ ;
```

using exactly the same sigma-matrix coefficients of Figure F.5. Other two cards must be added in order to use exactly the TRACE 3D R-matrix formalism:

```
16. 3. 1863.153; //proton mass, as ratio of electron mass
22. 0.05 0.0 700 0.0 /SPAC/ ; //space charge card
```

SBEND in TRACE 3D

The bending magnet definition in TRACE 3D requires: The edge angles are described with another type code and

Parameter	Value	Description
NT	8	Type code for bending
α [deg]	30	angle of bend in horizontal plane
ρ [mm]	250	radius of curvature of central trajectory
n	0	field-index gradient
vf	0	flag for vertical bending

Table F.1: Bending magnet description in TRACE 3D and values used in the simulation

parameters which include also the fringe field. They must be added before and after the bending magnet if entrance and exit edge angles are present and if the fringe field has to be taken into account. In particular for the entrance edge angle: A same configuration has been used for exit edge angle using $\beta = 5$ deg. The beam envelopes in the three phase planes for this simulation are shown in Figure F.6.

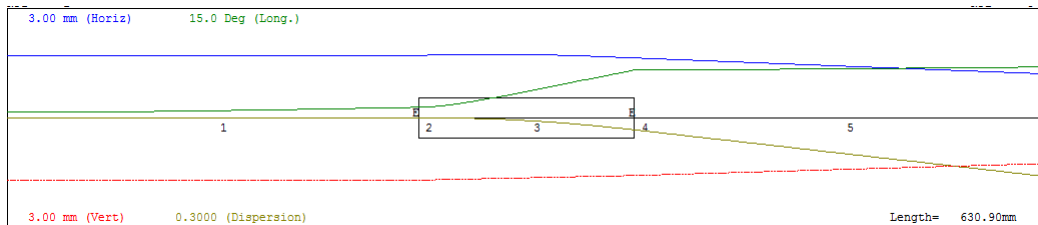


Figure F.6: Beam envelopes in TRACE 3D for a SBEND with entrance and exit edge angles. The blue line describes the beam envelope in the horizontal plane, the red line in the vertical plane, the green line in the longitudinal plane. The yellow line displays the dispersion

Parameter	Value	Description
NT	9	Type code for edge
β [deg]	10	pole-face rotation
ρ [mm]	250	radius of curvature of central trajectory
g [mm]	20	total gap of magnet
K_1	0.36945	fringe-field factor
K_2	0.36945	fringe-field factor

Table F.2: Edge angle description in TRACE 3D and values used in the simulation

Parameter	Value	Description
Card	4	Type code for bending
L [m]	30	Effective length of the central trajectory
B_0 [kG]	250	Central field strength
n	0	field-index gradient

Table F.3: Bending magnet description in TRANSPORT and values used in the simulation

SBEND in TRANSPORT

The bending magnet definition in TRANSPORT requires: As for TRACE 3D, the edge angles are described with another card and parameters. In TRANSPORT, however, the fringe field is not automatically included with the edge angle, but it is described by a own card as reported in the Table F.4. Running the Graphic TRANSPORT version, the beam envelopes in the transverse phase planes for this simulation are shown in Figure F.7.

Beam size and emittance comparison

In the next table, the results of the comparison between TRACE 3D and TRANSPORT in terms of the transversal beam sizes at the end of each element in the line are summarised. The perfect agreement between these two codes arises immediately looking at Figure F.8. The same comparison has been performed in terms of horizontal and longitudinal emittance, both expressed in π -mm-mrad. While the vertical emittance remains constant and equal to the initial value ($\varepsilon_y = 0.730 \pi$ -mm-mrad), the horizontal and longitudinal emittances are expected growing after the bending magnet. The results are reported in Table F.6 and in Figure F.9.

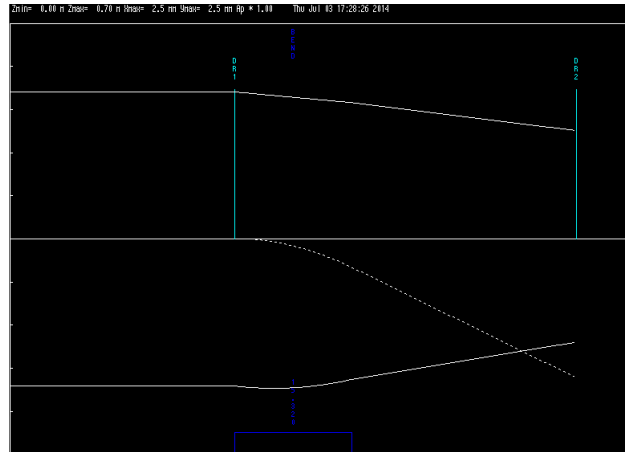


Figure F.7: Beam envelopes in TRANSPORT for a SBEND with entrance and exit edge angles. The continuous lines describe the beam envelope in the vertical plane (above) and horizontal plane (below). The dashed line displays the dispersion.

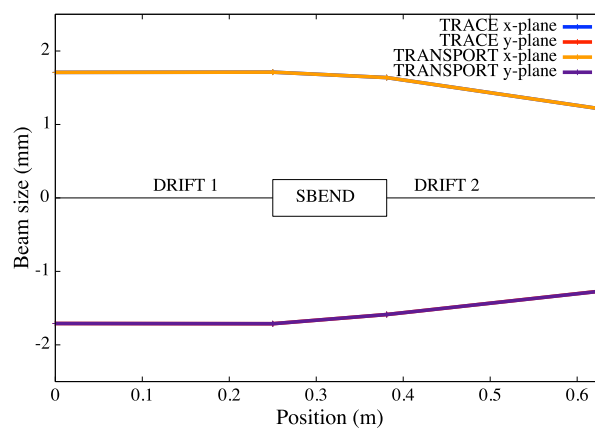


Figure F.8: Transversal beam size comparison between TRACE 3D and TRANSPORT

Parameter	Value	Description
Card	2	Type code for edge
β [deg]	10	pole-face rotation
Card	16	Type code for fringe field
g [mm]	10	half-gap of magnet
K_1	0.36945	fringe-field factor
K_2	0.36945	fringe-field factor

Table F.4: Edge angle and fringe field description in TRANSPORT and values used in the simulation

		TRACE 3D		TRANSPORT	
Position	z (m)	σ_x (mm)	σ_y (mm)	σ_x (mm)	σ_y (mm)
Input	0.000	1.709	1.709	1.709	1.709
Drift 1	0.250	1.712	1.712	1.712	1.712
Edge	0.250	1.712	1.712	1.712	1.712
Bend	0.381	1.638	1.587	1.638	1.587
Edge	0.381	1.638	1.587	1.638	1.587
Drift 2	0.631	1.206	1.264	1.206	1.264

Table F.5: Transversal beam size at the end of each element in the line printed out by TRACE 3D and TRANSPORT

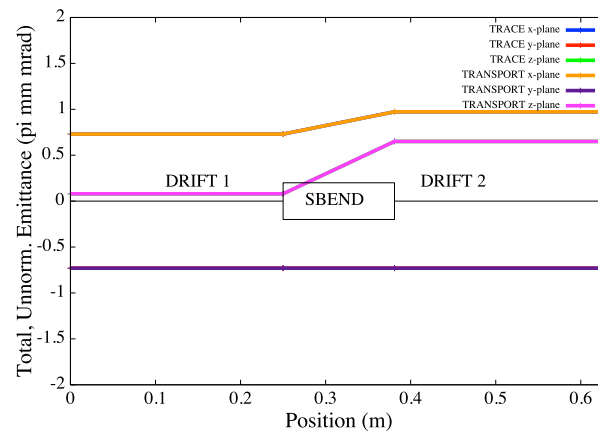


Figure F.9: Emittance comparison between TRACE and TRANSPORT

		TRACE 3D		TRANSPORT	
Position	z (m)	ε_x	ε_z	ε_x	ε_z
Input	0	0.730	0.08	0.730	0.08
Drift 1	0.250	0.730	0.08	0.730	0.08
Edge	0.250	0.730	0.08	0.730	0.08
Bend	0.381	0.973	0.65	0.973	0.65
Edge	0.381	0.973	0.65	0.973	0.65
Drift 2	0.631	0.973	0.65	0.973	0.65

Table F.6: Horizontal and longitudinal emittance comparison between TRACE 3D and TRANSPORT, both expressed in π -mm-mrad

From TRACE 3D to TRANSPORT

Parameter	Trace 3D	Transport
Bend card	8	4
Angle	Input parameter [deg]	Output information [deg]
Magn. field	Calculated. [T]	Input parameter [kG]
Radius of curv.	Input parameter [mm]	Output information [m]
Field-index	Input parameter	Input parameter
Effect. length	Calculated [mm]	Input parameter [m]
Edge card	9	2
Edge angle	Input parameter [deg]	Input parameter [deg]
Vertical gap	9	16.5
Gap	Total [mm]	Half-gap [cm]
Fringe field card	9	16.7 / 16.8
K_1	Default: 0.45	Default: 0.5
K_2	Default: 2.8	Default: 0
Bend direction	Bend angle sign	Coord. rotation
Horiz. right	Angle > 0	Angle > 0
Horiz. left	Angle < 0	Card 20
Vertical bend	Card 8, vf > 0	Card 20

Table F.7: Bending magnet features in TRACE 3D and TRANSPORT

F.1.8 Relations to OPAL-T

In OPAL, the beam dynamics approach (time integration) is hence completely different from the envelope-like supported by TRACE 3D and TRANSPORT. The three codes support different units and require diverse parameters for the input beam. A summary of their main features is reported in Table F.8.

Code	TRACE 3D	TRANSPORT	OPAL
Type	Envelope	Envelope	Time integration
Input	Twiss, Emittance	Sigma, Momentum	Sigma, Energy
Units	mm-mrad, deg-keV	cm-rad, cm-%	m- $\beta\gamma$

Table F.8: Main features of the three beam dynamics codes: TRACE 3D, TRANSPORT and OPAL

F.1.9 OPAL-T Units

OPAL-T supports the following internal coordinates and units for the three phase planes:

- **horizontal plane:**
X [m] horizontal position of a particle relative to the axis of the element;
PX [$\beta_x\gamma$] horizontal canonical momentum;
- **vertical plane:**
Y [m] vertical position of a particle relative to the axis of the element;
PY [$\beta_y\gamma$] horizontal canonical momentum;
- **longitudinal plane:**
Z [m] longitudinal position of a particle in floor-coordinates;
PZ [$\beta_z\gamma$] longitudinal canonical momentum;

F.1.10 OPAL-T Input beam

For the input beam, a GAUSS distribution type has been chosen. For transferring the TRANSPORT (or TRACE 3D) input beam in terms of sigma-matrix coefficients, it necessary to:

- adjust the units: from mm to m;
- correct for the factor $\sqrt{5}$: from total to RMS distribution;
- multiply for the relativistic factor $\beta\gamma = 0.1224$ for 7 MeV protons;

In case of the modified sigma-matrix in Figure F.5, the corresponding OPAL parameters for the GAUSS distributions are:

T3D SIGMA	OPAL -T
1.7088 mm	SIGMAX = 1.7088/sqrt(5)e-3 m
0.4272 mrad	SIGMAPX = 0.4272/sqrt(5)*0.1224e-3
1.7088 mm	SIGMAY = 1.7088/sqrt(5)e-3 m
0.4272 mrad	SIGMAPY = 0.4272/sqrt(5)*0.1224e-3
0.1092 mm	SIGMAZ = 0.1092/sqrt(5)e-3 m
0.0717 %	SIGMAPZ = (0.0717*10)/sqrt(5)*0.1224e-3

At the end of this calculation, the input beam in OPAL is:

```
D1: DISTRIBUTION, DISTRIBUTION=GAUSS,
SIGMAX = 0.7642e-03, SIGMAPX= 0.0234e-03, CORRX= 0.0,
SIGMAY = 0.7642e-03, SIGMAPY= 0.0234e-03, CORRY= 0.0,
SIGMAZ = 0.0488e-03, SIGMAPZ= 0.0392e-03, CORRZ= 0.0, R61= 0.0,
INPUTMOUNITS=NONE;
```

F.1.11 Comparison TRACE 3D and OPAL-T

In this section, the comparison between TRACE 3D and OPAL-T is discussed starting from SBEND definition in OPAL-T. The transport line described in section F.1.7 has been simulated in OPAL using 10.000 particles and 10^{-11} s time step. The bending magnet features of Table F.1 and F.2 have been transformed in OPAL language as:

```
Bend: SBEND, ANGLE = 30.0 * Pi/180.0,
      K1=0.0,
      E1=0, E2=0,
      FMAPFN = "1DPROFILE1-DEFAULT",
      ELEMEDGE = 0.250, // end of first drift
      DESIGNENERGY = 7E+06, // ref energy eV
      L = 0.1294,
      GAP = 0.02;
```

- **SBEND without edge angles:**

```
// Bending magnet configuration:
K1=0.0,
E1=0, E2=0,
```

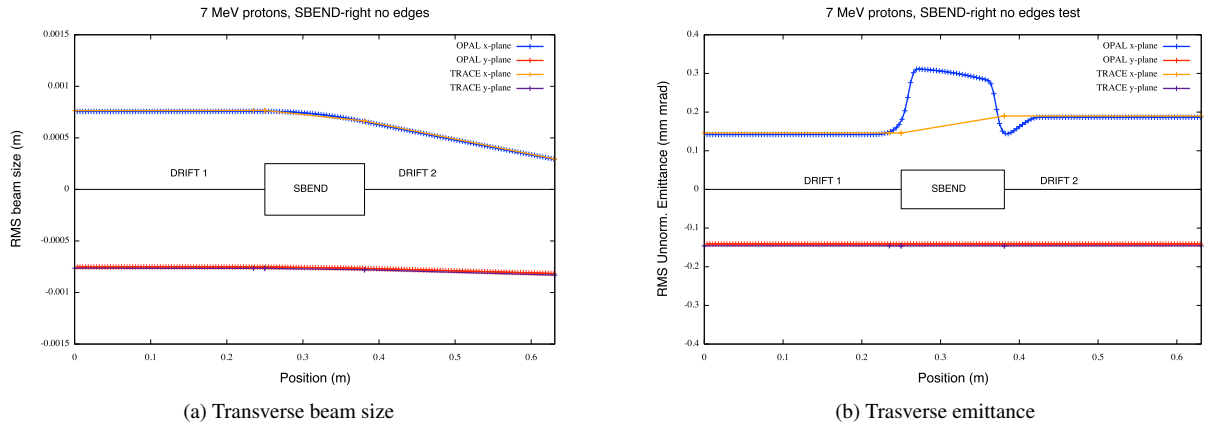


Figure F.10: TRACE 3D and OPAL comparison: SBEND without edge angles

A good overall agreement has been found between the two codes in term of beam size and emittance. The different behaviour inside the bending magnet for the horizontal emittance is still undergoing study and it's probably due to a diverse coordinate system in the two codes.

- **SBEND with edge angles:**

```
// Bending magnet configuration:
K1=0.0,
E1=10*Pi/180.0, E2=5* Pi/180.0,
```

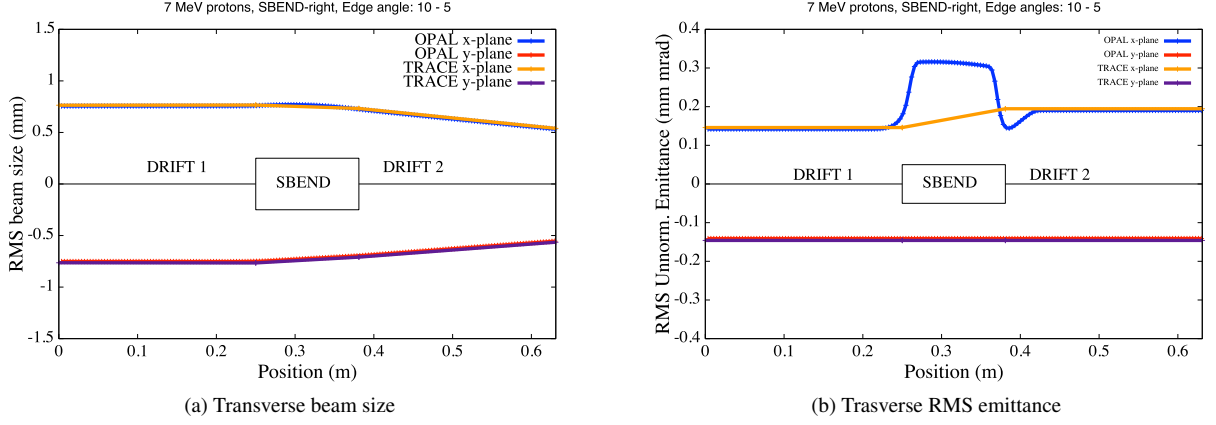


Figure F.11: TRACE 3D and OPAL comparison: SBEND with edge angles

Even in this case, a good overall agreement has been found between the two codes in term of beam size and emittance.

- **SBEND with field index:**

The field index parameter $K1$ is defined as:

$$K1 = \frac{1}{B\rho} \frac{\partial B_y}{\partial x}, \quad (F.4)$$

(see §8.4.1). Instead, in TRACE 3D the field index parameter n is:

$$n = -\frac{\rho}{B_y} \frac{\partial B_y}{\partial x}. \quad (F.5)$$

In order to have a significative focusing effect on both transverse planes, the transport line has been simulated in TRACE 3D using $n = 1.5$. Since, a different definition exists between OPAL and TRACE 3D on the field index, the n -parameter translation in OPAL language has been done with the following test:

TEST 1: $K1 = n/\rho^2$
 TEST 2: $K1 = n$
 TEST 3: $K1 = n/\rho$

Only the TEST 2 reports a reasonable behaviour on the beam size and emittance, as shown in Figure F.12 using:

```
// Bending magnet configuration:
K1=1.5
E1=0, E2=0,
```

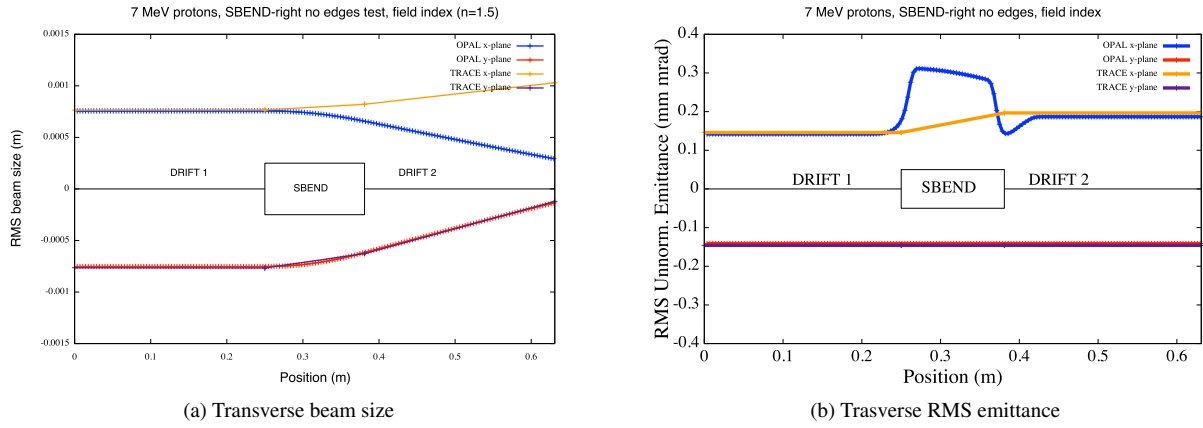


Figure F.12: TRACE 3D and OPAL comparison: SBEND with field index and default field map

Concerning the emittances and vertical beam size, a perfect agreement has been found, instead a defocusing effect appears in the horizontal plane. These results have been obtained with the default field map provided by OPAL. However, a better result, only in the beam size as shown in Figure F.13, is achieved using a test field map in which the fringe field extension has been changed in the thin lens approximation.

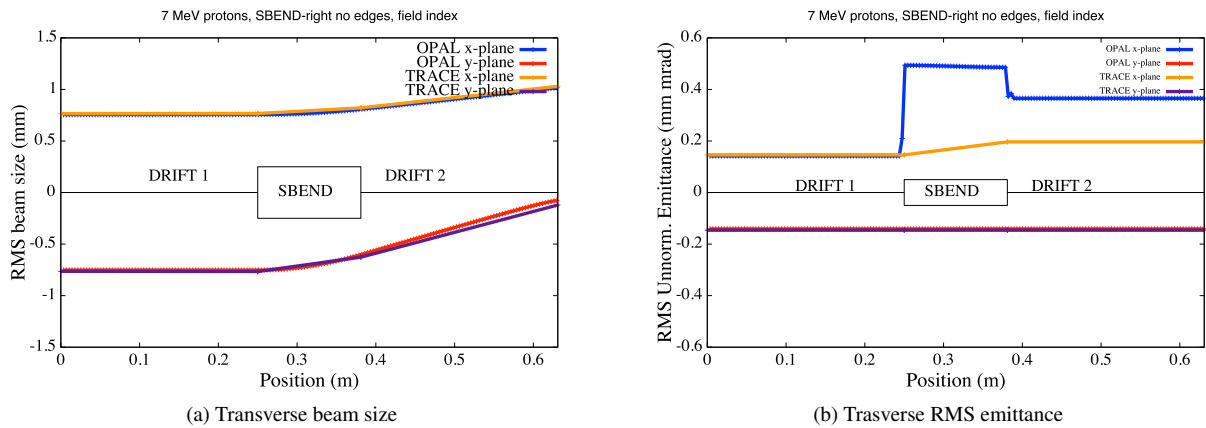


Figure F.13: TRACE 3D and OPAL comparison: SBEND with field index and test field map

From TRACE 3D to OPAL-T

Parameter	Trace 3D	OPAL-T
Bend card	8	SBEND or RBEND
Angle	Input parameter [deg]	Input/Calc. parameter [rad]
Magn. field	Calculated. [T]	Input/Calc parameter [T]
Radius of curv.	Input parameter [mm]	Output information [m]
Field-index	Input parameter	Input parameter
Length	Calculated [mm]	Input/Calc parameter [m]
Length type	Effective	Straight
Edge card	9	SBEND or RBEND
Edge angle	Input parameter [deg]	Input parameter [rad]
Vertical gap	9	SBEND or RBEND
Gap	Total [mm]	Total [m]
Fringe field card	9	FIELD MAP
K_1	Default: 0.45	-
K_2	Default: 2.8	-
OPAL-T Bend direction	Bend angle sign	Coord. rotation
Horiz. right	Angle > 0	Angle > 0
Horiz. left	Angle < 0	Angle < 0
Vertical bend	Card 8, vf > 0	Coord. rotation

Table F.9: Bending magnet features in TRACE 3D and OPAL-T

F.1.12 Conclusion

- **TRACE 3D and TRANSPORT:**

- a perfect agreement has been found between these two codes in transversal envelope and emittance;
- changing the TRANSPORT units, the input beam parameters, in terms of sigma-matrix coefficients, can directly be imported from TRACE 3D file.

- **TRACE 3D and OPAL-T:**

- a good agreement has been found between these two codes in case of sector bending magnet with and without edge angles;
- the default magnetic field map seems not working properly if the field index is not zero
- an improvement of the test map used is needed in order to match the TRACE 3D emittance (see Figure F.13).

F.2 Hard Edge Dipole Comparison with ELEGANT

F.2.1 OPAL Dipole

When defining a dipole (SBEND or RBEND) in OPAL a fringe field map which defines the range of the field and the Enge coefficients is required. If no map is provided, the code uses a default map. Here is a dipole definition using the default map:

Example:

```
bend1: SBEND, ANGLE = bend_angle,
      E1 = 0, E2 = 0,
      FMAPFN = "1DPROFILE1-DEFAULT",
      ELEMEDGE = drift_before_bend,
      DESIGNENERGY = bend_energy,
      L = bend_length,
      WAKEF = FS_CSR_WAKE;
```

Please refer to the OPAL manual C.11 for the definition of the field map and the default map 1DPROFILE1-DEFAULT. It defines a fringe field that extends to 10 cm away from a dipole edge in both directions and it has both B_y and B_z components. This makes the comparison between OPAL and other codes which uses a hard edge dipole by default, cumbersome because one needs to carefully integrate through the fringe field region in OPAL in order to come up with the integrated fringe field value (FINT in ELEGANT) that usually used by these codes, e.g. the ELEGANT and the TRACE3D. So we need to find a default map for the hard edge dipole in OPAL.

F.2.2 Map for Hard Edge Dipole

The proposed default map for a hard edge dipole can be:

```
1DProfile1 0 0 2
-0.00000001 0.0 0.00000001 3
-0.00000001 0.0 0.00000001 3
-99.9
-99.9
```

On the first line, the two zeros following 1DProfile1 are the orders of the Enge coefficient for the entrance and exit edge of the dipole. 2cm is the default dipole gap width. The second line defines the fringe field region of the entrance edge of the dipole which extends from -0.00000001cm to 0.00000001cm . The third line defines the same fringe field region for the exit edge of the dipole. The 3s on both line don't mean anything, they are just placeholders. On the fourth and fifth line, the zeroth order Enge coefficients for both edges are given. Since they are large negative numbers, the field in the fringe field region has no B_z component and its B_y component is just like the field in the middle of the dipole. Figure F.14 compares the emittances and beam sizes obtained by using the hard edge map, the default map and the ELEGANT. One can see that the results produced by the hard edge map match the ELEGANT results when FINT is set to zero.

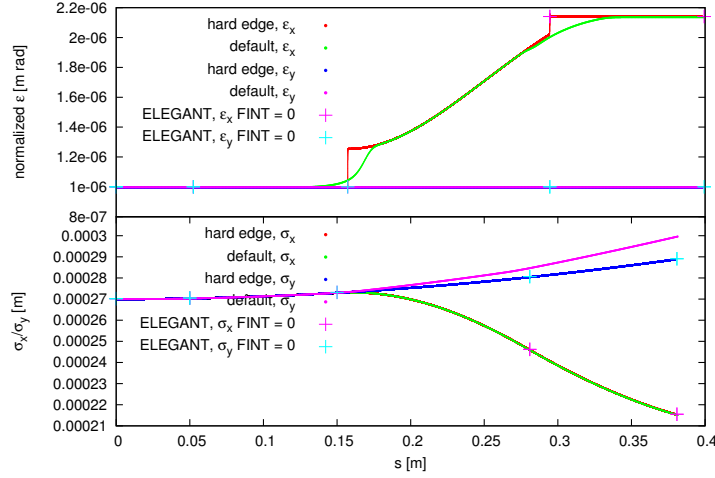


Figure F.14: Compare emittances and beam sizes obtained by using the hard edge map (OPAL), the default map (OPAL), and the ELEGANT

F.2.3 Integration Time Step

When the hard edge map is used for a dipole, finer integration time step is needed to ensure the accurate of the calculation. Figure F.15 compares the normalised emittances generated using the hard edge map in OPAL with varying time steps to those from the ELEGANT. $0.01ps$ seems to be a optimal time step for the fringe field region. To speed up the simulations, one can use larger time steps outside the fringe field regions. In Figure F.15, one can observe a discontinuity in the horizontal emittance when the hard edge map is used in the calculation. This discontinuity comes from the fact that OPAL emittance is calculated at an instant time. Once the beam or part of the beam gets into the dipole, its P_x gets a kick which will result in a sudden emittance change.

Figure F.16 and F.17 examine the effects of the fringe field range and the integration time step on the simulation accuracy. Figure F.17 is a zoom-in plot of Figure F.16. We can conclude that the size of the integration time step has more influence on the accuracy of the simulation.

F.3 1D CSR comparison with ELEGANT

1D-CSR wakefunction can now be used for the drift element by defining its attribute `WAKEF = FS_CSR_WAKE`. In order to calculate the CSR effect correctly, the drift has to follow a bending magnet whose CSR calculation is also turned on.

Example:

```
bend1: SBEND, ANGLE = bend_angle,
      E1 = 0, E2 = 0,
      FMAPFN = ``1DPROFILE1-DEFAULT``,
      ELEMEDGE = drift_before_bend,
      DESIGNENERGY = bend_energy,
      L = bend_length,
      WAKEF = FS_CSR_WAKE;

drift1: DRIFT, L=0.4, ELEMEDGE = drift_before_bend +
      bend_length, WAKEF = FS_CSR_WAKE;
```

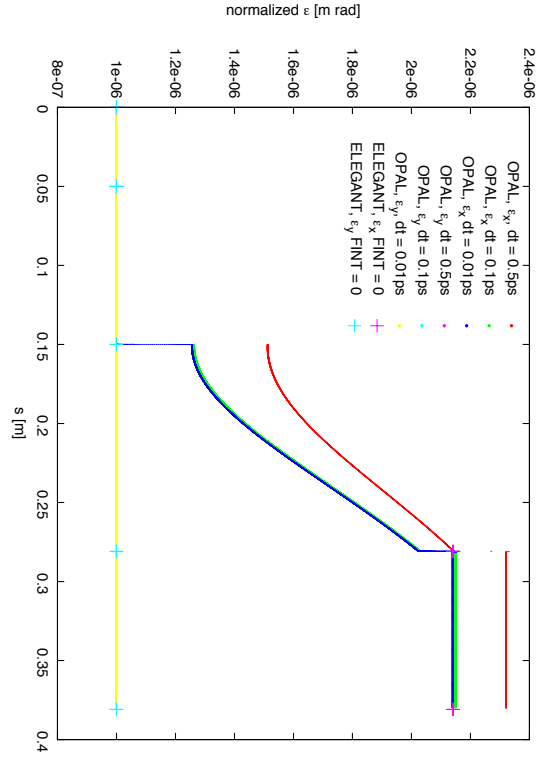


Figure F.15: Horizontal and vertical normalised emittances for different integration time steps

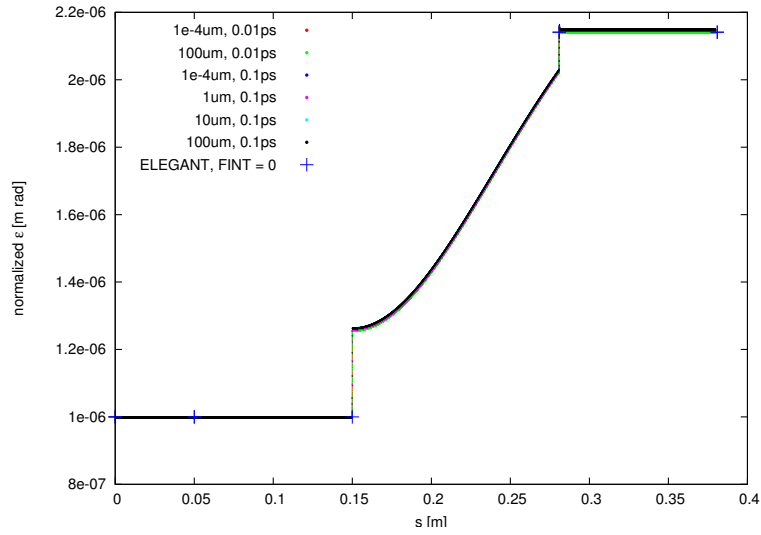


Figure F.16: Normalised horizontal emittance for different fringe field ranges and integration time steps

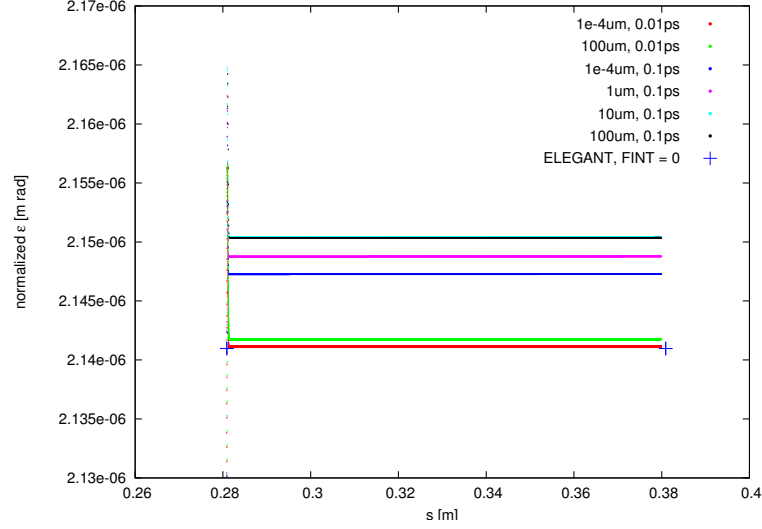


Figure F.17: Zoom in on the final emittance in Figure F.17

F.3.1 Benchmark

The OPAL dipoles all have fringe fields. When comparisons are done between OPAL and ELEGANT [66] for example, one needs to appropriately set the FINT attribute of the bending magnet in ELEGANT in order to represent the field correctly. Although ELEGANT tracks in the $(x, x', y, y', s, \delta)$ phase space, where $\delta = \frac{\Delta p}{p_0}$ and p_0 is the momentum of the reference particle, the watch point output beam distributions from the ELEGANT are list in $(x, x', y, y', t, \beta\gamma)$. If one wants to compare ELEGANT watch point output distribution to OPAL unit conversion needs to be performed, i.e.

$$\begin{aligned} P_x &= x' \beta \gamma, \\ P_y &= y' \beta \gamma, \\ s &= (\bar{t} - t) \beta c. \end{aligned}$$

To benchmark the CSR effect, we set up a simple beamline with 0.1m drift + 30 degree sbend + 0.4m drift. When the CSR effect is turn off, Fig. F.18 shows that the normalized emittances calculated using both OPAL and ELEGANT agree. The emittance values from OPAL are obtained from the .stat file, while for ELEGANT, the transverse emittances are obtained from the sigma output file (enx, and eny), the longitudinal emittance is calculated using the watch point beam distribution output.

When CSR calculations are enabled for both the bending magnet and the following drift, Fig. F.19 shows the average δ or $\frac{\Delta p}{p}$ change along the beam line, and Fig. F.20 compares the normalized transverse and longitudinal emittances obtained by these two codes. The average $\frac{\Delta p}{p}$ can be found in the centroid output file (Cdelta) from ELEGANT, while in OPAL, one can calculate it using $\frac{\Delta p}{p} = \frac{1}{\beta^2} \frac{\Delta \bar{E}}{\bar{E} + mc^2}$, where $\Delta \bar{E}$ is the average kinetic energy from the .stat output file. In the drift space following the bending magnet, the CSR effects are calculated using Stupakov's algorithm with the same setting in both codes. The average fractional momentum change $\frac{\Delta p}{p}$ and the longitudinal emittance show good agreements between these codes. However, they produce different horizontal emittances as indicated in Fig. F.20.

One important effect to notice is that in the drift space following the bending magnet, the normalized emittance $\varepsilon_x(x, P_x)$ output by OPAL keeps increasing while the trace-like emittance $\varepsilon_x(x, x')$ calculated by ELEGANT does not. This can be explained by the fact that with a relatively large energy spread (about 3% at the end of the dipole due to CSR), **an correlation** between transverse position and energy can build up in a drift thereby induce emittance growth. However, this effect can only be observed in the normalized emittance calculated

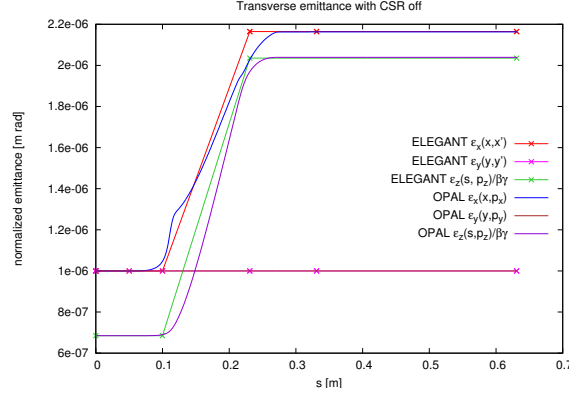


Figure F.18: Comparison of the trace space using ELEGANT and OPAL

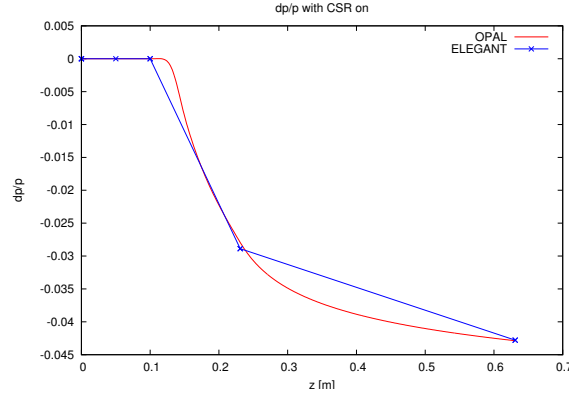
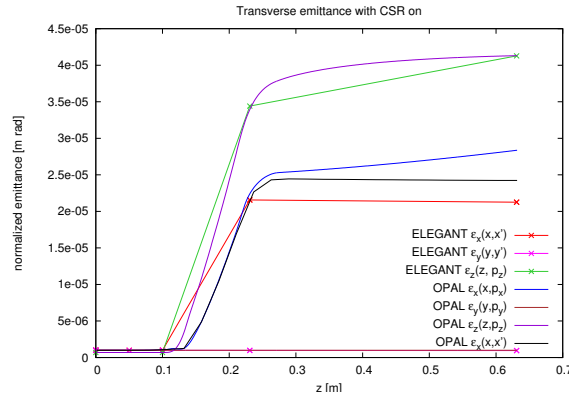
Figure F.19: $\frac{\Delta p}{p}$ in Elegant and OPAL

Figure F.20: Transverse emittances in ELEGANT and OPAL

with $\varepsilon_x(x, P_x) = \sqrt{\langle x^2 \rangle \langle P_x^2 \rangle - \langle x P_x \rangle^2}$ where $P_x = \beta \gamma x'$, not the trace-like emittance which is calculated as $\varepsilon_x(x, x') = \beta \gamma \sqrt{\langle x^2 \rangle \langle x'^2 \rangle - \langle x x' \rangle^2}$ [67]. In Fig. F.20, a trace-like horizontal emittance is also calculated for the OPAL output beam distributions. Like the ELEGANT result, this trace-like emittance doesn't grow in the drift. However, their differences come from the ELEGANT's lack of CSR effect in the fringe field region.

F.4 OPAL & IMPACT-T

impactt

This benchmark compares rms quantities such as beam size and emittance of OPAL and IMPACT-T [39, 40, 41]. A **cold** 10 mA H⁺ bunch is expanding in a 1 meter drift space. A Gaussian distribution, with a cut at 4σ is used. The charge is computed by assuming a 1 MHz structure i.e. $Q_{\text{tot}} = \frac{I}{\nu_{\text{rf}}}$. For the simulation we use a grid with 16^3 grid point and open boundary condition. The number of macro particles is $N_p = 10^5$.

F.4.1 OPAL Input

```

OPTION, ECHO = FALSE, PSDUMPFREQ = 10,
STATDUMPFREQ = 10, REPARTFREQ = 1000,
PSDUMPLocalFRAME = FALSE;

TITLE, string="Gaussian bunch drift test";

Edes      = 0.001;           // GeV
CURRENT = 0.01;             // A

gamma=(Edes+PMASS)/PMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*PMASS;

D1: DRIFT, ELEMEDGE = 0.0, L = 1.0;

L1: LINE = (D1);

Fs1: FIELDSOLVER, FSTYPE = FFT, MX = 16, MY = 16, MT = 16, BBOXINCR=0.1;

Dist1: DISTRIBUTION, DISTRIBUTION = GAUSS,
      OFFSETX = 0.0, OFFSETY = 0.0, OFFSETZ = 15.0e-3,
      SIGMAX = 5.0e-3, SIGMAY = 5.0e-3, SIGMAZ = 5.0e-3,
      OFFSETPX = 0.0, OFFSETPY = 0.0, OFFSETPZ = 0.0,
      SIGMAPX = 0.0, SIGMAPY = 0.0, SIGMAPZ = 0.0,
      CORRX = 0.0, CORRY = 0.0, CORRZ = 0.0,
      CUTOFFX = 4.0, CUTOFFY = 4.0, CUTOFFLONG = 4.0;

Beam1: BEAM, PARTICLE = PROTON, CHARGE = 1.0, BFREQ = 1.0e6, PC = P0,
      NPART = 1E5, BCURRENT = CURRENT, FIELDSOLVER = Fs1;

SELECT, LINE = L1;

TRACK, LINE = L1, BEAM = Beam1, MAXSTEPS = 1000, ZSTOP = 1.0, DT = 1.0e-10;
  RUN, METHOD = "PARALLEL-T", BEAM = Beam1, FIELDSOLVER = Fs1, DISTRIBUTION = Dist1;
ENDTRACK;
STOP;

```

F.4.2 IMPACT-T Input

```
!Welcome to \impactt\ input file.
```

```

!All comment lines start with "!" as the first character of the line.
! col row
1 1
!
! information needed by the integrator:
! step-size, number of steps, and number of bunches/bins (??)
!
!   dt      Ntstep  Nbunch
1.0e-10    700      1
!
! phase-space dimension, number of particles, a series of flags
! that set the type of integrator, error study, diagnostics, and
! image charge, and the cutoff distance for the image charge
!
! PSdim Nptcl  integF  errF  diagF  imchgF  imgCutOff (m)
6 100000  1 0 1 0 0.016
!
! information about mesh: number of points in x, y, and z, type
! of boundary conditions, transverse aperture size (m),
! and longitudinal domain size (m)
!
!   Nx  Ny  Nz  bcF   Rx    Ry    Lz
16 16 16 1 0.15 0.15 1.0e5
!
!
! distribution type number (2 == Gauss), restart flag, space-charge substep
! flag, number of emission steps, and max emission time
!
! distType  restartF  substepF  Nemission  Temission
2           0         0         -1          0.0
!
!   sig*   sigp*  mu*p*  *scale  p*scale  xmu*      xmu*
!
0.005 0.0 0.0  1. 1. 0.0 0.0
0.005 0.0 0.0  1. 1. 0.0 0.0
0.005 0.0 0.0  1. 1. 0.0 0.0462
!
! information about the beam: current, kinetic energy, particle
! rest energy, particle charge, scale frequency, and initial cavity phase
!
! I/A   Ek/eV      Mc2/eV          Q/e  freq/Hz  phs/rad
0.010   1.0e6      938.271998e+06  1.0  1.0e6    0.0
!
!
! ===== machine description starts here =====
! the following lines, which must each be terminated with a '//',
! describe one beam-line element per line; the basic structure is
! element length, ???, ???, element type, and then a sequence of
! at most 24 numbers describing the element properties
!   0  drift tube      2          zedge radius
!   1  quadrupole     9          zedge, quad grad, fileID,

```

```

!                                     radius, alignment error x, y
!                                     rotation error x, y, z
! L/m  N/A N/A  type  location of starting edge  v1  <B0><B0><B0>  v23  /
1.0    0    0    0    0.0                      0.5                /

```

F.4.3 Results

A good agreement is shown in the Figures F.21 and F.22. This proves to some extent the compatibility of the space charge solvers of OPAL and IMPACT-T.

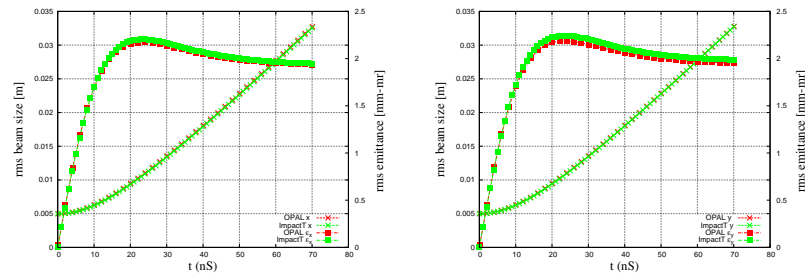


Figure F.21: Transverse beam sizes and emittances in IMPACT-T and OPAL

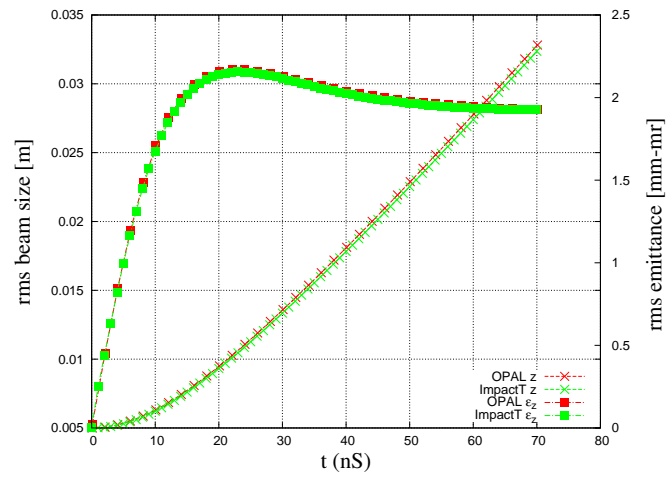


Figure F.22: Longitudinal beam size and emittance in IMPACT-T and OPAL

Appendix G

Changelog

G.1 Changes in OPAL Version 1.3.0

G.1.1 Cyclotron Element

In order to accomodate axial injection through a spiral inflector, the Cyclotron element has been updated with a z coordinate and a pz for the reference particle (ZINIT, PZINIT).

G.1.2 AMG Solver

The AMG Solver (including arbitrary boundary condition) is enabled per default.

G.1.3 Variable Frequency RF

Variable frequency RF is now available in OPAL-CYCL mode when using the RINGDEFINITION as described in Section 8.13. A new LOCAL_CARTESIAN_OFFSET element is introduced to enable overlapping elements in the RINGDEFINITION, described in Section 8.11.

G.1.4 Appendix

Two new Appendixes, Appendix F with Benchmarks (TRANSPORT, TRACE3D and IMPACT-T) and in Appendix E a description of the auto phasing mechanism in OPAL-T and OPAL-E is available.

G.1.5 Unit Tests

Parts of OPAL can be tested using unit tests, details can be found at A.3

G.2 Changes in OPAL Version 1.2.0

G.2.1 Distribution

The distribution class is totally rewritten. You have to indicate the units to be used for p_x , p_y and p_z with the new argument INPUTMOUNITS. At the moment electron volt (EV) or $\beta\gamma$ (NONE which is default) are available.

```
Dist1:DISTRIBUTION, DISTRIBUTION = GAUSS,  
                      SIGMAX = 1.e-03,  
                      SIGMAPX = 1.0,
```

```

CORRX = 0.0,
SIGMAY = 1.e-03,
SIGMAPY = 1.0,
CORRY = 0.0,
SIGMAT = 2.e-3 ,
INPUTMOUNITS = EV;

```

An hands-on example is given in Sec. 3.4. Furthermore the `PC` argument of the `BEAM` command defines the mean energy of the generated distribution. The attributes `PZ` and `PT` are no longer supported. The full correlation matrix σ can be defined for the `GAUSS` distribution.

G.2.2 SBEND and RBEND

The elements `SBEND` and `RBEND` for `OPAL-T` have also been rewritten. (See Section 8.4 for a full description.) If you have input files with an `SBEND` or `RBEND` already defined, here are the key changes you must make for those definitions to work properly:

1. All angles are now specified in radians.
2. Energy units for the bend reference particle are now specified in eV.
3. For bend definitions that use your own, custom field map, *do not specify the bend length*, `L`.

G.2.3 Misalignment

The misalignment argument `DZ`, misalignment in `z` direction is no longer supported, instead use `ELEMEDGE`.

G.2.4 New elements: SBend3D & RINGDEFINITION

`SBend3D` (Sec. 8.4.6) & `RingDefinition` (8.11) adding FFAG and more general Ring modeling capabilities.

G.2.5 Space Charge Solver

Still in an experimental stage, two iterative solvers can be enabled (see A). With both solvers the exact boundary conditions can be applied. Per default and in the binary distribution the solvers are disabled. We are working on regression tests and plan to have them ready in Version 1.2.1.

Bibliography

- [1] *The Graphical Kernel System (GKS)*. ISO, Geneva, July 1985. International Standard ISO 7942.
- [2] B. Autin and Y. Marti. *Closed Orbit Correction of Alternating Gradient Machines using a small Number of Magnets*. CERN/ISR-MA/73-17, CERN, 1973.
- [3] D.P. Barber, K. Heinemann, H. Mais and G. Ripken, *A Fokker–Planck Treatment of Stochastic Particle Motion within the Framework of a Fully Coupled 6-dimensional Formalism for Electron-Positron Storage Rings including Classical Spin Motion in Linear Approximation*, DESY report 91-146, 1991.
- [4] R. Bartolini, A. Bazzani, M. Giovannozzi, W. Scandale and E. Todesco, *Tune evaluation in simulations and experiments*, CERN SL/95-84 (AP) (1995).
- [5] J. D. Bjorken and S. K. Mtingwa. *Particle Accelerators* **13**, pg. 115.
- [6] E.M. Bollt and J.D. Meiss, *Targeting chaotic orbits to the Moon through recurrence*, Phys. Lett. **A204**, 373 (1995).
- [7] P. Bramham and H. Henke. private communication and LEP Note LEP-70/107, CERN.
- [8] Karl L. Brown. *A First-and Second-Order Matrix Theory for the Design of Beam Transport Systems and Charged Particle Spectrometers*. SLAC 75, Revision 3, SLAC, 1972.
- [9] Karl L. Brown, D. C. Carey, Ch. Iselin, and F. Rothacker. *TRANSPORT — A Computer Program for Designing Charged Particle Beam Transport Systems*. CERN 73-16, revised as CERN 80-4, CERN, 1980.
- [10] A. Chao. *Evaluation of beam distribution parameters in an electron storage ring*. Journal of Applied Physics, 50:595–598, 1979.
- [11] A. W. Chao and M. J. Lee. *SPEAR II Touschek lifetime*. SPEAR-181, SLAC, October 1974.
- [12] M. Conte and M. Martini. *Particle Accelerators* **17**, 1 (1985).
- [13] E. D. Courant and H. S. Snyder. *Theory of the alternating gradient synchrotron*. Annals of Physics, 3:1–48, 1958.
- [14] M. Donald and D. Schofield. *A User’s Guide to the HARMON Program*. LEP Note 420, CERN, 1982.
- [15] A. Dragt. *Lectures on Nonlinear Orbit Dynamics, 1981 Summer School on High Energy Particle Accelerators, Fermi National Accelerator Laboratory, July 1981*. American Institute of Physics, 1982.
- [16] D. A. Edwards and L. C. Teng. *Parametrisation of linear coupled motion in periodic systems*. IEEE Trans. on Nucl. Sc., 20:885, 1973.
- [17] M. Giovannozzi, *Analysis of the stability domain of planar symplectic maps using invariant manifolds*, CERN/PS 96-05 (PA) (1996).

- [18] H. Grote. *GXPLOT User's Guide and Reference Manual*. LEP TH Note 57, CERN, 1988.
- [19] LEP Design Group. *Design Study of a 22 to 130 GeV e^+e^- Colliding Beam Machine (LEP)*. CERN/ISR-LEP/79-33, CERN, 1979.
- [20] M. Hanney, J. M. Jowett, and E. Keil. *BEAMPARAM — A program for computing beam dynamics and performance of e^+e^- storage rings*. CERN/LEP-TH/88-2, CERN, 1988.
- [21] R. H. Helm, M. J. Lee, P. L. Morton, and M. Sands. *Evaluation of synchrotron radiation integrals*. IEEE Trans. Nucl. Sc., NS-20, 1973.
- [22] F. James. *MINUIT, A package of programs to minimise a function of n variables, compute the covariance matrix, and find the true errors*. program library code D507, CERN, 1978.
- [23] E. Keil. *Synchrotron radiation from a large electron-positron storage ring*. CERN/ISR-LTD/76-23, CERN, 1976.
- [24] D. E. Knuth. *The Art of Computer Programming*. Volume 2, Addison-Wesley, second edition, 1981. Semi-numerical Algorithms.
- [25] J. Laskar, C. Froeschlé and A. Celletti, *The measure of chaos by the numerical analysis of the fundamental frequencies. Application to the standard mapping*, Physica D**56**, 253 (1992).
- [26] H. Mais and G. Ripken, *Theory of Coupled Synchro-Betatron Oscillations*. DESY internal Report, DESY M-82-05, 1982.
- [27] M. Meddahi, *Chromaticity correction for the $108^\circ/60^\circ$ lattice*, CERN SL/Note 96-19 (AP) (1996).
- [28] J. Milutinovic and S. Ruggiero. *Comparison of Accelerator Codes for a RHIC Lattice*. AD/AP/TN-9, BNL, 1988.
- [29] B. W. Montague. *Linear Optics for Improved Chromaticity Correction*. LEP Note 165, CERN, 1979.
- [30] G. Ripken, *Untersuchungen zur Strahlführung und Stabilität der Teilchenbewegung in Beschleunigern und Storage-Ringen unter strenger Berücksichtigung einer Kopplung der Betatronschwingungen*. DESY internal Report R1-70/4, 1970.
- [31] F. Ruggiero, *Dynamic Aperture for LEP 2 with various optics and tunes*, Proc. Sixth Workshop on LEP Performance, Chamonix, 1996, ed. J. Poole (CERN SL/96-05 (DI), 1996), pp. 132–136.
- [32] L. C. Teng. *Concerning n -Dimensional Coupled Motion*. FN 229, FNAL, 1971.
- [33] U. Völkel. *Particle loss by Touschek effect in a storage ring*. DESY 67-5, DESY, 1967.
- [34] R. P. Walker. *Calculation of the Touschek lifetime in electron storage rings*. 1987. Also SERC Daresbury Laboratory preprint, DL/SCI/P542A.
- [35] P. B. Wilson. *Proc. 8th Int. Conf. on High-Energy Accelerators*. Stanford, 1974.
- [36] A. Wrulich and H. Meyer. *Life time due to the beam-beam bremsstrahlung effect*. PET-75-2, DESY, 1975.
- [37] Birdsall, Langdon. *Plasma Physics via computer simulation*. Page 340-341
- [38] G. Fubiani, J. Qiang, E. Esarey, W. P. Leemans, G. Dugan. *Space charge modeling of dense electron beams with large energy spreads*. Accelerators and Beams, 9, 064402 (2006)
- [39] J. Qiang, S. Lidia, R. D. Ryne, and C. Limborg-Deprey *A Three-Dimensional Quasi-Static Model for High Brightness Beam Dynamics Simulation* Lawrence Berkeley National Laboratory. Paper LBNL-59098. <http://repositories.cdlib.org/lbnl/LBNL-59098>

- [40] J. Qiang, S. Lidia, R. D. Ryne, C. Limborg-Deprey, *Three-Dimensional Quasi-Static Model for High Brightness Beam Dynamics Simulation* Phys. Rev. Special Topics - Accel. Beams 9, 044204, (2006).
- [41] J. Qiang, S. Lidia, R. D. Ryne, C. Limborg-Deprey, *Erratum: Three-Dimensional Quasi-Static Model for High Brightness Beam Dynamics Simulation* Phys. Rev. Special Topics - Accel. Beams 10, 129901, (2007).
- [42] W. Joho private discussions
- [43] J. H. Billen, L. M. Young *POISSON SUPERFISH*. LA-UR-96-1834, Los Alamos National Laboratory, 2004
- [44] J. E. Spencer, H. A. Enge *Split-Pole Magnetic Spectrograph for Precision Nuclear Spectroscopy* Nucl. Instr. Meth. 49 (1967) 181-193
- [45] J.M. Sanz-Sern and M.P. Calvo *Numerical Hamiltonian Problems* Chapman and Hall 1994
- [46] E. Forest. Phys. Lett. A 158, p99, 1991
- [47] P. Arbenz, R. Geus and S. Adam *Solving Maxwell eigenvalue problems for accelerating cavities*, Physical Review Special Topics - Accelerators and Beams, 4, pp. 022001-1:022001-10, 2001.
- [48] P. Arbenz, M. Becka, R. Geus, U. Hetmaniuk and T. Mengotti *On a Parallel Multilevel Preconditioned Maxwell Eigensolver*, PARCO, 32(2), pp. 157-165, 2006, doi: 10.1016/j.parco.2005.06.005
- [49] K. Flöttmann, *Note on the thermal emittance of electrons emitted by Cesium Telluride photo cathodes*, TESLA FEL-Report, 1997-01.
- [50] J. E. Clendenin, T. Kotseroglou, G. A. Mulhollan, D. T. Palmer, J. F. Schmerge, *Reduction of thermal emittance of RF guns* NIM A, 455 (2000) 198-201.
- [51] D. H. Dowell and J. F. Schmerge, *Quantum efficiency and thermal emittance of metal photocathodes* Phys. Rev. STAB, 12 (2009) 074201.
- [52] A Adelmann and P Arbenz and Y Ineichen, *A Fast Parallel Poisson Solver on Irregular Domains Applied to Beam Dynamic Simulations 0907.4863v1 arXiv* (2009)
- [53] Y. Feng and J. P. Verboncoeur, *Transition from Fowler-Nordheim field emission to space charge limited current density* Phys. Plasmas, 13, 073105 (2006)
- [54] R. H. Fowler and L. Nordheim, *Electron Emission in Intense Electric Fields* Proc. R. Soc. London, Ser. A 119, 173 (1928)
- [55] J. H. Han, *Dynamics of Electron Beam and Dark Current in Photocathode RF Guns* PhD Thesis, Desy, 2005 Available Online on <http://www-library.desy.de/preparch/desy/thesis/desy-thesis-05-045.pdf>
- [56] C. Wang and A. Adelmann and Y. Ineichen, *A Field Emission and Secondary Emission Model in OPAL* Proc. of HB2010, MOPD55 (2010)
- [57] M. A. Furman and M. T. F. Pivi, *Probabilistic model for the simulation of secondary electron emission* Phys. Rev. ST Accel. Beams, 5, 124404 (2002)
- [58] J. Rodney M. Vaughan, *A New Formula for Secondary Emission Yield* IEEE Trans. on Electron Devices, 36:9, 1963 (1989)
- [59] J. Rodney M. Vaughan, *Secondary Emission Formulas* IEEE Trans. on Electron Devices, 40:4, 830 (1993)
- [60] C. Vicente and M. Mattes and D. Wolk and H. L. Hartnagel and J. R. Mosig and D. Raboso, *FEST3D - A Simulation Tool for Multipactor Prediction* Proc. of MULCOPIM 2005, (2005)

- [61] S. Anza and C. Vicente and J. Gil and V. E. Boria and B. Gimeno and D. Raboso, *Nonstationary Statistical Theory for Multipactor Phys. Plasmas*, 17, 062110 (2010)
- [62] A. Henderson, *ParaView Guide: A Parallel Visualization Application Kitware Inc.*, (2007)
- [63] L. Serafini and J. B. Rosenzweig, *Phys. Rev. E* **55** (1996).
- [64] M. Ferrario, J. E. Clendenin, D. T. Palmer, J. B. Rosenzweig, L. Serafini, *HOMDYN Study for the LCLS RF Photo-Injector*, SLAC-PUB-8400, March, 2000
- [65] arxiv
- [66] M. Borland, "Elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation", Advanced Photon Source LS-287, September 2000.
- [67] K. Floettmann, *PRSTAB*, 6, 034202 (2003)
- [68] K. Crandall and D. Rusthoi, *Documentation for TRACE: An Interactive Beam-Transport Code*, Los Alamos National Laboratory
- [69] PSI Graphic Transport Framework by U. Rohrer based on a CERN-SLAC-FERMILAB version by K.L. Brown et al.
- [70] http://aea.web.psi.ch/Urs_Rohrer/MyWeb/moregifts/gantryb.gif
- [71] <http://amas.web.psi.ch/people/aadelmann/pub/www/>
- [72] F. J. Sacherer, *RMS Envelope Equations with Space Charge*, CERN, Geneva, Switzerland
- [73] R. W. Hockney, *Methods Comput. Phys.* 9, 136-210 (1970).
- [74] J. W. Eastwood and D. R. K. Brownrigg, *J. Comp. Phys.*, 32, 24-38 (1979).
- [75] R. W. Hockney and J. W. Eastwood, "Computer Simulation using Particles," Taylor & Francis Group (1988).
- [76] J. Qiang and R. Ryne, "Parallel 3D Poisson solver for a charged beam in a conducting pipe," *Comp. Phys. Comm.* 138, 18-28 (2001).
- [77] J. Qiang, R. L. Gluckstern, "Three-dimensional Poisson solver for a charged beam with large aspect ratio in a conducting pipe," *Comp. Phys. Comm.* 160 (2) (2004) 120–128.
- [78] R. D. Ryne, "A new technique for solving Poisson's equation with high accuracy on domains of any aspect ratio," ICFA Beam Dynamics Workshop on Space-Charge Simulation, Oxford, April 2-4, 2003.
- [79] D. T. Abell, P. J. Mullowney, K. Paul, V. H. Ranjbar, J. Qiang, R. D. Ryne, "Three-Dimensional Integrated Green Functions for the Poisson Equation," THPAS015, Proc. 2007 Particle Accelerator Conference (2007).
- [80] J. Qiang, S. Lidia, R. Ryne, C. Limborg-Deprey, *Phys. Rev. ST Accel. Beams* 9, 044204 (2006), and *Phys. Rev. ST Accel. Beams* 10, 129901(E) (2007).
- [81] R. D. Ryne et al., "Recent progress on the MaryLie/IMPACT Beam Dynamics Code," Proc. 2006 International Computational Accelerator Physics Conference, Chamonix, France, 2-6 Oct 2006. (2006).
- [82] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, "Numerical Recipes: The Art of Scientific Computing," Cambridge University Press (2007).
- [83] J. D. Jackson, "Classical Electrodynamics," John Wiley & Sons (1998).

- [84] William R. Smythe, "Static and Dynamic Electricity," problems for Chapter 5, Hemisphere Publishing Corp. (1989).
- [85] J. Qiang, "A high-order fast method for computing convolution integral with smooth kernel," *Comp. Phys. Comm.* 18, 313316, (2010).
- [86] R. D. Ryne, S. Habib, and T. P. Wangler, "Halos of Intense Proton Beams," *Proc. 1995 Particle Accelerator Conference* (1995).
- [87] M. Ferrario, J. E. Clendenin, D. T. Palmer, J. B. Rosenzweig, L. Serafini, *HOMDYN Study for the LCLS RF Photo-Injector*, SLAC-PUB-8400, March, 2000
- [88] M. Ferrario, *HOMDYN User Guide*